# eHint: An Efficient Protocol for Uploading Small-Size IoT Data

Tsung-Yen Chan, Yi Ren, Yu-Chee Tseng, and Jyh-Cheng Chen
Department of Computer Science, National Chiao Tung University, Taiwan
Emails:{tychan, renyi, yctseng, jcc}@cs.nctu.edu.tw

***Abstract*—IoT (Internet of Things) has attracted a lot of attention recently. IoT devices need to report their data or status to base stations at various frequencies. The IoT communications observed by a base station normally exhibit the following characteristics: (1) massively connected, (2) lightly loaded per packet, and (3) periodical or at least mostly predictable. The current design principals of communication networks, when applied to IoT scenarios, however, do not fit well to these requirements. When a large number of devices contend to send small packets, the signaling overhead is not cost-effective. To address this problem, our previous work [1] proposes the Hint protocol, which is slot-based and schedule-oriented for uploading IoT devices' data. In this work, we extend [1] to support data transmissions for multiple resource blocks. We assume that the uplink payloads from IoT devices are small, each taking very few slots (or resource blocks), but devices are massive. The main idea is to "encode" information in a tiny broadcast that allows each device to "decode" its transmission slots, thus significantly reducing transmission overheads and contention overheads. Our simulation results verify that the protocol can significantly increase channel utilization compared with traditional schemes.**

***Index Terms*—Communication Protocol, Internet of Things (IoT), Machine-Type Communication (MTC), Wireless Network.**

## I. Introduction

Internet of Things (IoT) is a new paradigm that has become very popular in both research community and industry recently [2]. IoT devices range from small tags and sensors to more complicated actuators and machines. Through these various things, we are able to closely connect the cyber world with the physical world. According to [2], the number of IoT devices may reach 50 billions in the next decade. Such massively connected Internet devices may make significant impacts on mobile networks' traffics [3]. Statistics show that 50% of IoT packets are less than 100 bytes [4], [5].

To support collecting small data from massive IoT devices, the wireless network architecture should be carefully redesigned. Currently, the data collection approaches for IoT devices fall into two categories: (1) connection-oriented and (2) connectionless. The connection-oriented approaches [6]–[11], for example, need to allocate channels or time slots to User Equipment (UEs) in advance. For example, in Long-Term Evolution (LTE), a Radio Resource Control (RRC) connection establishment procedure includes more than 12 interactions in the Radio Access Network (RAN) side and 15 interactions in the Core Network (CN) side, no matter how much data needs to be transmitted. According to [12], one bit of small data costs 5-70 times more signaling traffic compared to one bit of streaming data.

For connection-oriented solutions, a successful transmission needs to go through three phases: (1) random access, (2) signaling, and (3) data transmission. To increase transmission efficiency, connectionless solutions skip the connection setup procedure for infrequent small data transmissions [12]–[15]. In these approaches, devices transmit small data right after the random access procedure. That is, the small data traffic is piggy-backed with control messages. This means that the transmission happens in the control-plane, violating the design principle of separating of user-plane and control-plane.

In this paper, we observe that IoT communications at a Base Station (BS) normally exhibit the following characteristics: (1) massively connected, (2) lightly loaded per packet, and (3) periodical or at least mostly predictable. Improved based on our previous work, the Hint protocol [1], we further propose an enhanced Hint protocol, eHint, which can allocate *multiple slots* to a group of devices. The basic idea is to broadcast a seed and a vector that allow devices to decode their slot allocation easily. As a result, not only the random-access cost is largely eliminated, but also the signaling cost is minimized. The broadcast cost is much lower than 1-to-1 notification in typical approaches. We demonstrate through extensive simulations that the eHint significantly increases channel utilization over traditional schemes.

The rest of this paper is organized as follows. Related works are reviewed in Section II. Section III formally defines our problem. Section IV introduces our eHint protocol, followed by performance evaluation results in Section V. Section VI draws our conclusions.

## II. Related Work

Connection-oriented network architecture has been intensively studied [6]–[11]. To establish a connection in LTE, UEs utilize random-access (RA) procedures to obtain radio resources for data transmissions. When there are too many UEs contending for the limited RA preambles, it may lead to significant collision and congestion, resulting in long delays [16]. Reference [6] proposes to split the available RA preambles into two disjoint subsets (one subset for MTC devices and the other for human-to-human devices) to deal with the heavy RA loading caused by massive MTC devices. In [7], a new contention-based uplink channel access for MTC devices is proposed, by which UEs can select their transmission opportunities randomly without indications from eNB, thus reducing latency and saving signaling overhead significantly. Considering multiple eNBs, [8] tries to optimize the Access Class Barring (ACB) [17] parameters based on congestion levels. In [9], a new RA scheme is proposed by taking advantage of the fixed uplink timing alignment. It achieves low collision probability, short access delay, and high energy efficiency, but it is only suitable for static devices. Reference [10] proposes to increase the amount of available contention space by expanding to the code domain. Reference [11] tries to decrease the RRC Connected-to-Idle transition time by reducing the RRC Inactivity Timer. The advantage is to save devices' power consumption. However, it may lead to high signaling overhead due to frequent RRC transitions when data inter-arrival time is short.

On the other hand, some research has considered connectionless approaches [12]–[15]. Reference [13] proposes a simultaneous radio access mechanism for asynchronous small data packet transmission. But this work only considers the performance of data detection. In [12], a slim RRC state is proposed to reduce the costs of RRC setup/release/maintenance, but it neglects the energy cost. To reduce the RA collision probability in the RA process and the signaling overhead, [14] proposes to directly piggyback small data to the exchange messages of the RA process. However, these solutions violate the design principle of separating of control and user planes and may have scalability concerns at the control plane. In [15], a combinatorial model and a fast approximation of the average number of successful users in a one-shot random access are proposed in a finite-user multi-channel slotted ALOHA system. But the impact of bursty arrival traffic to the RA procedure is neglected in this work.

Our recent work [1] proposes the Hint protocol, the first ID-free data transmission protocol that utilizes common hash functions as an agreement between a base station and massive IoT devices. As a result, even ID is not included in an uplink packet, the base station is still able to track the packet, reducing signaling cost significantly for massive connections. However, it only uses one resource block for each device's data transmission. In this paper, we improve [1] to support data transmissions for multiple resource blocks.

## III. Problem Statement

We consider a set $D = \{d_1, d_2, \cdots, d_m\}$ of $m$ IoT devices covered by a base station $BS$. Each IoT device $d_i$ needs to report its data at a regular pattern to the BS. We consider the problem of allocating radio resources for these devices to transmit their data to the BS. We make the following assumptions:

- The number of devices is massive in the sense that $m$ is quite large.
- Each device switches between two modes, *active* and *sleep*. When it intends to transmit data, it goes to the active mode; otherwise, it switches to the sleep mode.
- The active pattern of device $d_i$, $i = 1..m$, is denoted by a binary periodical function $P_i(t)$, where $t$ is a frame counter maintained by the BS. $P_i(t) = 1$ 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 every 3 minutes, then $P_i(t)$ can be a periodical function of period = 3 minutes. If one more humidity sensor which needs to report every 5 minutes is attached to $d_i$, then $P_i(t)$ is a function of the combination of two periodical functions with periods = 3 and 5 minutes, respectively.
- Whenever $P_i(t) = 1$, device $d_i$ needs to send $n_i$ slots of data to the BS at $t$. Considering the property of IoT applications, the value of $n_i$ is quite small (such as less than 3 or 5 slots).
- When $d_i$ registers into the BS, it should inform the BS its $P_i(t)$ and $n_i$. Therefore, the BS knows when and how much data $d_i$ will transmit data to it.
- Devices could be mobile and thus the set $D$ may change over time. However, the BS always knows the members of $D$.

To solve the radio resource allocation problem of these devices, we divide the communication channel into a sequence of fixed-length *frames*. Each frame consists of three parts: (1) Broadcast Part (`Bcast`): It is for the BS to broadcast and announce resource allocation information to devices. (2) Allocated Part (`Alloc`): It consists of multiple slots for devices to safely transmit their data to the BS without carrying their IDs. (3) Random Part (`Rand`): It is for any unscheduled/unpredicted transmission not arranged in `Alloc`. Our goal is to design an efficient access protocol for such transmission needs.

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. 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. Their requirements are $n_1 = 2$, $n_2 = 1$, and $n_3 = 3$ slots, respectively. The BS should schedule their transmissions in `Alloc`, through the announcement in `Bcast`. Our protocol does not guarantee all devices to get their slots in `Alloc`. If there is any requirement that can not be scheduled in `Alloc`, devices can use `Rand`. Also, any emergency traffic can use `Rand`.

## IV. The Enhanced Hint Protocol: eHint

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 (i) in `Bcast`. Then, each device can use the broadcast information and its ID to retrieve the transmission slots allocated to it through the function. The broadcast information in (i) is expected to be more efficient than typical 1-by-1 notifications. The protocol may not guarantee each device to get its transmission slot in `Alloc`, in which case devices can use traditional random access to contend in `Rand`.

Let $v$ be a vector and $h(s, ID)$ be a hash function. Here, $< s, v >$ is the broadcast information, where $s$ is a value computed by the BS and $v$ is a vector which indicates whether

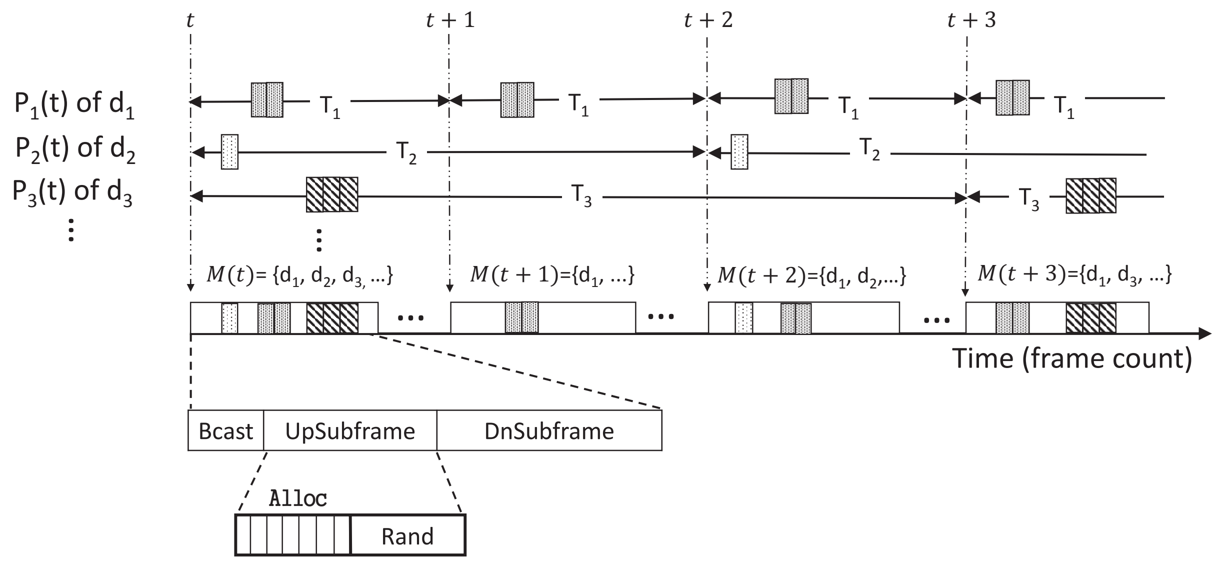

Fig. 1. The proposed frame structure of our Hint protocol [1]. $T_1$, $T_2$, and $T_3$ are the transmission periods of devices $d_1$, $d_2$, and $d_3$, respectively.

a device can transmit or not. Recall the active $P_i(t)$ of $d_i$. At the $t$-th frame, the $BS$ can compute the set of devices $M(t)$ that intend to transmit:

$$M(t) = \{d_i | P_i(t) = 1\}. \tag{1}$$

The maximal required slots in $t$ is thus

$$n_{req} = \sum_{d_i \in M(t)} n_i. \tag{2}$$

We set the length of $v$ to $n_{vec} \geq n_{req}$. The $i$-th element of $v$ (resp., `Alloc` ) is denoted by $v[i]$ (resp., `Alloc`$[i]$ ). The value of each $v[i]$ is in $\{0, 1, 2\}$. The vector $v$ is for devices to decode whether their hashed slots are safe for transmission or not. Specially, if it is safe to transmit in $n_i$ continuous slots in `Alloc`, the corresponding elements in $v$ will be

$$1\underbrace{2\cdots2}_{n_i-1}$$

where "1" means "starting slot" and "2" means "continuous slot". If it is not safe to transmit or a slot is not assigned to any device, a "0" is used. Note that, although the length of `Alloc` is not announced explicitly, it will be implied by the numbers of 1s and 2s in $v$.

At the $t$-th frame, the protocol works as follows:

1) The BS repeats the following steps a few times.
   - Randomly picks a seed $s$ and computes $h(s, d_i)\ mod\ n_{vec}$ for each device $d_i \in M(t)$.
   - Selects a subset $M' \subseteq M(t)$ of devices such that only devices in $M'$ can correctly decode $v$ for safe transmission.
2) The BS then selects the $s$ and the corresponding $M'$ from one of the iterations in Step 1 such that the total safe transmission slots is the largest.
3) After the selection, the BS encodes $v$ as follows:
   - (Collision-free) For each $d_i \in M'$ (i.e., allowed to transmit) such that $k = h(s, d_i)\ mod\ n_{vec}$, set $v[k] = 1$ and $v[k + 1 : k + n_i - 1] = 2$.
   - (Empty/Collision) The rest of elements of $v$ are all set to 0.
4) The BS broadcasts $< s, v >$ in `Bcast` to all devices.
5) For each device $d_i \in M(t)$, it receives $< s, v >$ in `Bcast` and then computes $k = h(s, d_i)\ mod\ n_{vec}$. To check whether $d_i$ is allowed to transmit or not, there are two cases:
   - Case of $n_i = 1$: If $v[k] = 1$ and $v[k + 1] = 0$ or 1, $d_i$ is allowed to transmit; otherwise, $d_i$ can not transmit.
   - Case of $n_i \geq 2$: If $v[k] = 1$, $v[k+1 : k+n_i-1] = 2$, and $v[k + n_i] = 0/1$, $d_i$ is allowed to transmit; otherwise, it can not transmit.
6) For each device $d_i \in M(t)$ that determines (in Step 5) that it can transmit safely, $d_i$ uploads its data at `Alloc`$[j : j + n_i - 1]$, where $j$ is the number of 1s and 2s in $v[0 : k - 1]$ and $k = h(s, d_i)\ mod\ n_{vec}$. If $d_i$ finds itself to be not allowed to transmit, it can use random access to contend for transmission in `Rand`. (It is implied here that the length of `Alloc`, denoted by $n_{all}$, is the total number of 1s and 2s in $v$.)

Fig. 2(a) illustrates the above procedure in frame $t$. An example is shown in Fig. 2(b). There are 7 devices intending to transmit in frame $t$. These devices need 1~4 slots, as denoted by $n_i$. Hence, $n_{req} = 13$. The length of $v$ is also set to $n_{vec} = n_{req} = 13$. After hashing, $d_1$'s second data slot and $d_3$'s data slot are overlapped. Also, $d_1$'s fourth data slot and $d_2$'s first data slot are overlapped. Suppose that devices $d_1$, $d_4$, and $d_7$ are selected to transmit (i.e., $M' = \{d_1, d_4, d_7\}$). Then $v$ ="0122201201200" and the length of `Alloc` is $n_{all} = n_1 + n_4 + n_7 = 8$.

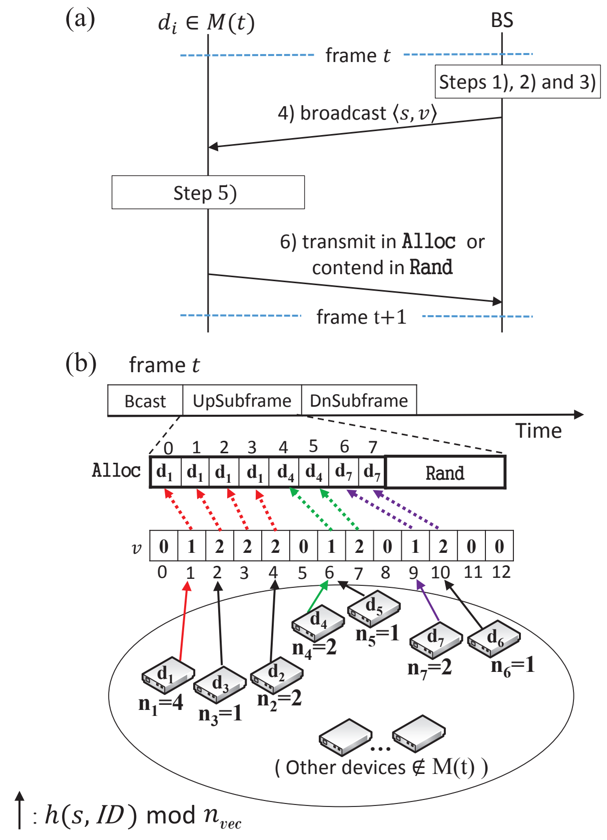

Fig. 2. (a) The message flow of the eHint protocol. (b) An example of 7 devices intending to transmit.

The encoding of $v$ is explained as follows. Since $d_1$ collides with $d_2$ and $d_3$, we have to choose either $d_1$ or $d_2$ and $d_3$ to transmit. Here, we choose $d_1$, so $v[1 : 4]$ ="1222". (If we choose $d_2$ and $d_3$ instead, $v[1 : 4]$ would become "0101".) Devices $d_4$ and $d_5$ have the same hashed position 6. Since we choose $d_4$ to transmit, $v[6 : 7]$="12". Since $d_5$ finds $v[7]$ ="2", it knows that it can not transmit. Device $d_6$ will not transmit because $v[10]$ ="2". Device $d_7$ finds that $v[9 : 11]$ = "120", so it knows that it is safe to transmit.

Then $d_1$ will transmit in $\texttt{Alloc}[0 : 3]$ because there is no 1 or 2 before $v[1]$. Similarly, $d_4$ and $d_7$ will transmit in $\texttt{Alloc}[4 : 5]$ and $\texttt{Alloc}[6 : 7]$ because there are 4 and 6 1s and 2s before $v[6]$ and $v[9]$, respectively.

We make two remarks below. First it is not hard to see that $n_{vec} \geq n_{req} \geq n_{all}$. A larger $n_{vec}$ makes vector $v$ longer and thus reduces the possibility of two devices colliding at the same position in $v$. The number of bits in $v$ is $2 \times n_{vec}$ and the increase of cost is low, but the benefit is more collision-free transmissions in Alloc (note that Step 6 ensures that there is no redundant slot in Alloc).

## V. Performance Evaluation

In this section, we evaluate the performance of the eHint protocol in terms of channel utilization through simulation. The results are compared with a traditional polling scheme which collects data from each device one-by-one (device address is assumed to be 64 bits). In the simulation, we assume that data size is small and randomly falls in the range of [1~3] ($1 \leq n_i \leq 3$) or [1~5] ($1 \leq n_i \leq 5$) slots. Unless otherwise specified, the following parameters are used: The total number devices in $M(t)$ (i.e., $|M(t)|$) is set to 20. Each slot is of length 2-20 byte. The simulation implements both an uplink random access procedure and our eHint protocol. One evaluation metric is *channel utilization* ($CU$):

$$CU = \frac{\text{payload}}{\text{payload + packet header}}. \tag{3}$$

In traditional polling schemes, we will add 64 bits of overhead for device address, which leads to

$$CU = \frac{\text{payload}}{\text{payload} + 64 \times |M(t)|}. \tag{4}$$

In our eHint protocol, the overhead is $< s, v >$, which leads to

$$CU = \frac{\text{payload}}{\text{payload} + s\text{-length} + 2 \times n_{vec}}. \tag{5}$$

Our simulation assumes $s$ to be 16 bits. Another evaluation metric is the number of safe transmissions that can be arranged in Alloc, which will be denoted by |Alloc| below.

*1) Impacts of $|M(t)|$:* In Fig. 3, we set $n_{vec} = 200$ and slot size to be 4 bytes. It compares $CU$ of our eHint protocol against the traditional polling scheme by varying $|M(t)|$. For the polling scheme, its $CU$ rate remains a constant value around 0.55 for [1~5] slots. The eHint protocol gives higher $CU$ as $|M(t)|$ increases. This validates that our eHint protocol can efficiently use broadcast to allocate resources to devices. As $|M(t)|$ reaches around 45, our $CU$ starts to saturate for [1~5] slots because we keep $n_{vec}$ a constant, which is reasonable. Fig. 4 shows the metric |Alloc| for the eHint protocol in different conditions. As the $|M(t)|$ increases, we can obtain more payload.

*2) Impacts of $n_{vec}$:* Fig. 5 and Fig. 6 demonstrate that $n_{vec}$ affects $CU$ and |Alloc|. Fig. 5 shows that $CU$ decreases slightly as the $n_{vec}$ increases (because the increasing rate of $n_{vec}$ is faster in denominator than payload in numerator of Eq. (5)). Fig. 6 shows that |Alloc| increases as the $n_{vec}$ increases because a larger $n_{vec}$ reduces the possibility of two devices colliding at the same position in $v$ and more transmission may be arranged.

*3) Impacts of slot size:* In Fig. 7 and Fig. 8, we show the impact of $CU$ and |Alloc| by varying slot size (2-20 bytes). Fig. 7 shows that our scheme still outperforms the polling scheme as slot size increases. For eHint, $CU$ actually reaches about 90% as slot size increases to 20 bytes. Fig. 8 shows that |Alloc| increases quite fast.

## VI. Conclusions

In this paper, we have proposed the eHint protocol for IoT small data transmission. In the eHint protocol, the BS only broadcasts a tiny hint, which informs devices: (1) if they are allowed to transmit in this frame and (2) how time slots are allocated to them. The hint message helps devices

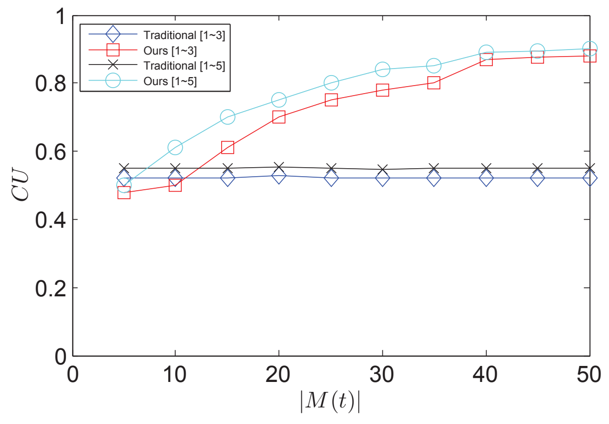

Fig. 3. $CU$ vs. $|M(t)|$.

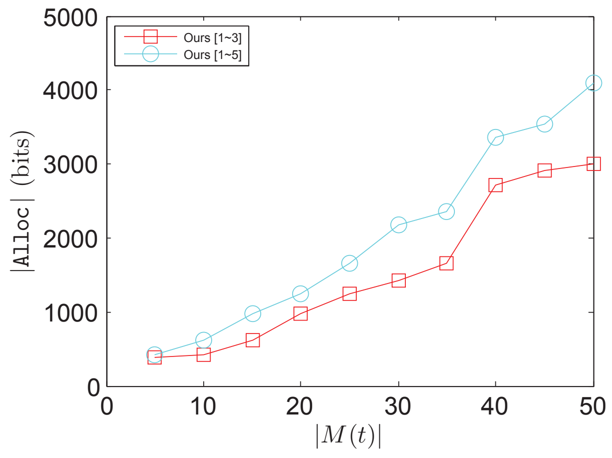

Fig. 4. The number of transmissions arranged in `Alloc` vs. $|M(t)|$.

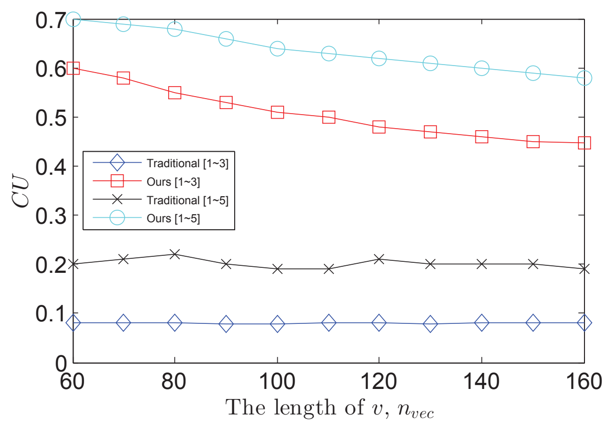

Fig. 5. $CU$ vs. $n_{vec}$.

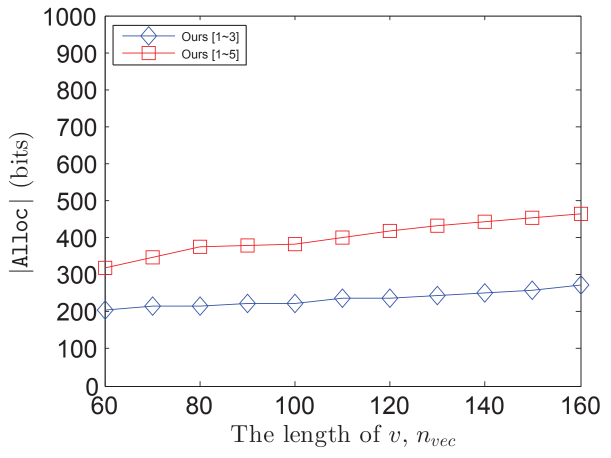

Fig. 6. The number of transmissions arranged in `Alloc` vs. $n_{vec}$.

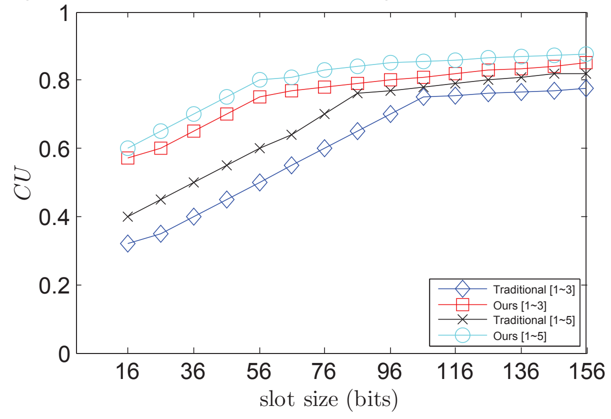

Fig. 7. $CU$ vs. slot size.

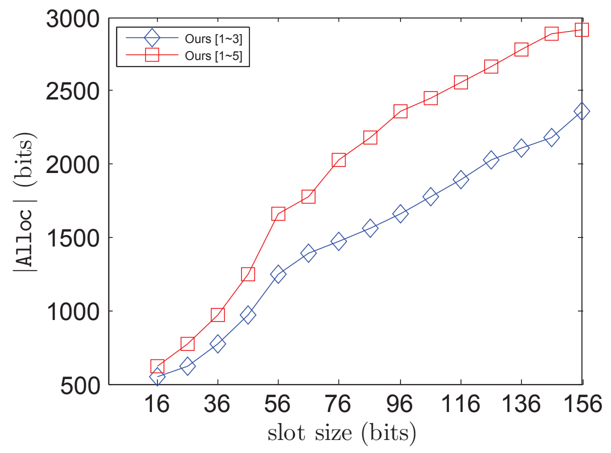

Fig. 8. The number of transmissions arranged in `Alloc` vs. slot size.

to skip the random access and contention procedures, thus reducing signaling cost significantly. The performance of the eHint protocol is compared with the traditional polling method and our simulation results validate our claims.

## Acknowledgement

This work is co-sponsored by MoE ATU Plan, MOST 105-2221-E-009-101-MY3, MOST 105-2221-E-009-100-MY3, MOST 105-2218-E-009-004, MOST 105-2218-E-009-008, MOST 104-3115-E-009-002, MOST 105-2218-E-009-003, Academia Sinica AS-102-TP-A06, ITRI, and Delta.

## References

[1] Y. Ren, R.-J. Wu, and Y.-C. Tseng, "The Hint protocol: Using a broadcast method to enable ID-free data transmission for dense IoT devices," in *Proc. PerCom Workshop, PerIoT '17*, Kona, Big Island, Hawaii, USA, Mar. 2017.

[2] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *CISCO white paper*, Apr. 2011.

[3] P. Cerwall, *Ericsson mobility report, on the pulse of the networked society, Mobile World Congress Edition*, Ericsson AB, Stockholm, Sweden, Tech. Rep., Nov. 2015.

[4] R. Sinha, C. Papadopoulos, and J. Heidemann, "Internet packet size distributions: Some observations," USC/Information Sciences Institute, Tech. Rep., May 2007.

[5] W. John and S. Tafvelin, "Analysis of internet backbone traffic and header anomalies observed," in *Proc. ACM SIGCOMM Conf. on Internet Measurement, IMC*, 2007, pp. 111–116.

[6] K.-D. Lee, S. Kim, and B. Yi, "Throughput comparison of random access methods for M2M service over LTE networks," in *Proc. IEEE GLOBECOM*, 2011, pp. 373–377.

[7] K. Zhou, N. Nikaein, R. Knopp, and C. Bonnet, "Contention based access for machine-type communications over LTE," in *Proc. IEEE Veh. Technol. Conf., VTC*, 2012, pp. 1–5.

[8] S.-Y. Lien, T.-H. Liau, C.-Y. Kao, and K.-C. Chen, "Cooperative access class barring for machine-to-machine communications," *IEEE Trans. Wireless Commun.*, vol. 11, no. 1, pp. 27–32, 2012.

[9] K. S. Ko, M. J. Kim, K. Y. Bae, D. K. Sung, J. H. Kim, and J. Y. Ahn, "A novel random access for fixed-location machine-to-machine communications in OFDMA based systems," *IEEE Commun. Lett.*, vol. 16, no. 9, pp. 1428–1431, 2012.

[10] N. K. Pratas, H. Thomsen, C. Stefanovic, and P. Popovski, "Code-expanded random access for machine-type communications," in *Proc. IEEE GLOBECOM Workshop*, 2012, pp. 1681–1686.

[11] S. C. Jha, A. T. Koc, M. Gupta, and R. Vannithamby, "Power saving mechanisms for M2M communication over LTE networks," in *Proc. 1st Int'l Black Sea Conf. on Communications and Networking, BlackSeaCom*, 2013, pp. 102–106.

[12] Y. Chen, G. Li, Z. Pan, and I. Chih-Lin, "Small data optimized radio access network signaling/control design," in *Proc. IEEE ICC*, 2014, pp. 49–54.

[13] C. S. Bontu, J. Ghimire, S. Periyalwar, and M. Pecen, "Asynchronous simultaneous small packet transmission in cellular wireless system," in *Proc. 16th. Int'l Symp. on Wireless Personal Multimedia Communications, WPMC*, 2013, pp. 1–5.

[14] W. Zou, Z. Li, and H. Wang, "Small data transmission at the detached machine-type-communication device," in *Proc. IEEE PIMRC*, 2012, pp. 13–17.

[15] C.-H. Wei, R.-G. Cheng, and S.-L. Tsao, "Modeling and estimation of one-shot random access for finite-user multichannel slotted ALOHA systems," *IEEE Commun. Lett.*, vol. 16, no. 8, pp. 1196–1199, 2012.

[16] A. Laya, L. Alonso, and J. Alonso-Zarate, "Is the random access channel of LTE and LTE-A suitable for M2M communications? A survey of alternatives," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 4–16, 2014.

[17] 3GPP TS36.331V10.5.0, *Evolved Universal Terrestrial Radio Access(E-UTRA); Radio Resource Control (RRC)*, 3GPP Std., Mar. 2012.