sTube: An Architecture for IoT Communication Sharing

Dan Wang, Wei Bao, Chuang Hu, Yi Qian, Muqiao Zheng, and Shi Wang

The requirements of IoT applications for different costs and data rates are significantly more diversified. The authors address this problem by proposing Sharing Tube (sTube), an architecture for IoT communication channel sharing. Intrinsically, a greater number of smart things can share a smaller number of dedicated channels. They present the design and a case implementation of sTube.

Abstract

In many IoT applications, there is a need to transmit the data collected by smart things to the cloud. The emerging SPS is one such system. Manufacturers/vendors have great interest in collecting data on their products for analysis and to enhance the overall smartness of the community. One obstacle is how to transmit the data from the things to the cloud. Rather than infiltrating other access networks for data transmission (e.g., a WiFi network in a building), a self-contained solution is preferred. It should be noted that many smart things are low-priced, making it impossible for them to afford the readily available channels (e.g., 3G/4G). To support this need, the industry is actively developing less expensive wireless communication channels. Unfortunately, the number of available choices in wireless communication channels is limited. However, the requirements of IoT applications for different costs and data rates are significantly more diversified. In this article, we address this problem by proposing Sharing Tube (sTube), an architecture for IoT communication channel sharing. Intrinsically, a greater number of smart things can share a smaller number of dedicated channels. We present the design and a case implementation of sTube. Our evaluation results demonstrate that sTube can achieve a cost saving of more than one order compared to a dedicated thing-to-cloud communication choice. This clears a core obstacle for many IoT applications in connecting to the cloud.

Introduction

One important value proposition of the Internet of Things (IoT) is the data generated by the things [1]. When sending such data to the cloud, with state-of-the-art data mining techniques and the computational power of the cloud, the added value can be significant [2]. For example, it has been shown that big building data (e.g., CO_2 data from heating, ventilation, and air conditioning [HVAC] systems) can be exploited to predict traffic status of nearby roads [3]. Smart post-sales (SPS), which is the case study in this article, is another typical example. Manufacturers/vendors of air conditioners and fans, elevators, and engine parts of automobiles are now transforming their machinery into smart machinery. When sending the data of their products to the cloud, the SPS can operate in trouble-preventing mode instead of troubleshooting mode — substantially improving the quality and reducing the cost of product maintenance. Moreover, manufacturers/vendors can learn the usage patterns of their products by their customers. Thus, they can recommend other products and develop top-up services based on such knowledge [4]. This can greatly enhance the overall smartness of the community.

To fully realize the aforementioned applications, the things should be accessible anywhere and anytime. One key question remains to be answered: how to transmit the data from the things to the cloud, in an easy-to-use and cost-effective way.

The manufacturers/vendors may develop a WiFi network for the IoT application. However, WiFi needs additional infrastructure, for example, a gateway that finally relays data to the cloud. This is not suitable for SPS applications. For example, a manufacturer/vendor would like to monitor all its air conditioners in a region, spanning a large number of buildings. The WiFi choice becomes infeasible since it needs deployment of WiFi on a building-by-building basis. If infiltrating existing WiFi networks in the buildings, the agreement will, again, be on a building-by-building basis. From the building's perspective, a typical building easily can have products from a large number of manufacturers/vendors. Building operators need to bear overwhelming liability if each manufacturer/vendor would like to infiltrate the existing WiFi network.

The manufacturer/vendor may rely on the infrastructure of an Internet service provider (ISP) and rent a dedicated wireless channel for each IoT device [5] to support the *thing-to-cloud communication* (TCC) links. This leads to a self-contained solution. However, current choices for thing-to-cloud communications are very limited. The readily available third/fourth generation (3G/4G) is clearly over-costly for the majority of IoT devices. The industry has realized this problem and is actively developing less costly wireless communication channels. User experience-category (CAT) represents a group of technologies with much smaller data rates and thus costs [6]. CAT1 was released in 2016, and CAT0 is under deployment [7]. Nevertheless, we may expect tens of choices of communication channels with different costs and data rates, but we will face hundreds, if not thousands, of heterogeneous requirements. In the SPS example, the cost of CAT1 might be fine for a chiller, but is too expensive for a fan.

Digital Object Identifier: 10.1109/MCOM.2018.1700515

Dan Wang, Chuang Hu, and Shi Wang are with Hong Kong Polytechnic Univ.; Wei Bao is with Sydney Univ.; Yi Qian is with Nebraska Lincoln Univ.; Muqiao Zheng is with Guangdong University of Technology.

We see a clear gap between the possible choices of wireless communication channels that can be used for thing-to-cloud communications, and the number of requirements on different costs and data rates from the IoT applications. To address this issue, we propose Sharing Tube (sTube), an architecture for IoT communication sharing. The objective of sTube is to organize a greater number of IoT devices with heterogeneous data communication requirements to *efficiently* share a smaller number of thing-to-cloud communication channels and transmit their data to the cloud. An illustration of an SPS application using sTube is shown in Fig. 1.

The contributions of this article can be summarized as:

- To the best of our knowledge, we are the first to clarify the necessity, scope, and exemplary applications of IoT communication sharing, and we discuss why existing architectures cannot meet the requirement.
- We articulate a TCC link sharing architecture problem (TCCS-arch) and present a solution, sTube, a modular architecture for IoT communication sharing.
- We prototype a fully functioning system for sTube. We evaluate sTube and show it can achieve substantial cost saving.

Background and Related Work

Communication Channels: Why We Need Sharing

The state-of-the-art wireless communication channels provide a variety of choices that trade off communication range, data rate, and costs for different application needs. However, the granularity of thing-to-cloud communication choices may not be enough, in the sense that for each IoT device with its own cost and data rate requirement, we can find a well-matched thing-to-cloud communication channel.

Readily available self-contained solutions (e.g., 3G/4G [8]) are provided by ISPs. 3G/4G is over-powerful and expensive for most IoT applications. Alternative solutions include LTE Category 1 (CAT1) released in 2016 and the soon-to-appear LTE Category 0 (CAT0). New choices are being developed, but the progress cannot match the surging requirements. More importantly, there may be requirements that have never been covered by ISPs. For example, CAT1 has a data rate of 5 Mb/s and a monthly cost of around $1 for a data volume of 40 MB. Assume there is an requirement where an equipment has a data volume of 45 MB but can only afford $1. ISPs will not deliberately develop such a plan since it makes CAT1 non-marketable. In a sharing environment, nearby equipment with residual data of 5 MB per month can be shared.

There are communication channels that are free but can only form a local (LOC) network. Short-range channels include Zigbee, Bluetooth, and so on. They are good for device-to-device communication. WiFi, LoRa, and SigFox [9] can provide longer-range wireless access. These are not self-contained since gateways are needed to reach the cloud outside. In our design, IoT devices will form LOC networks so as to share the TCC links. This article, however, will not emphasize the design of LOC networks.

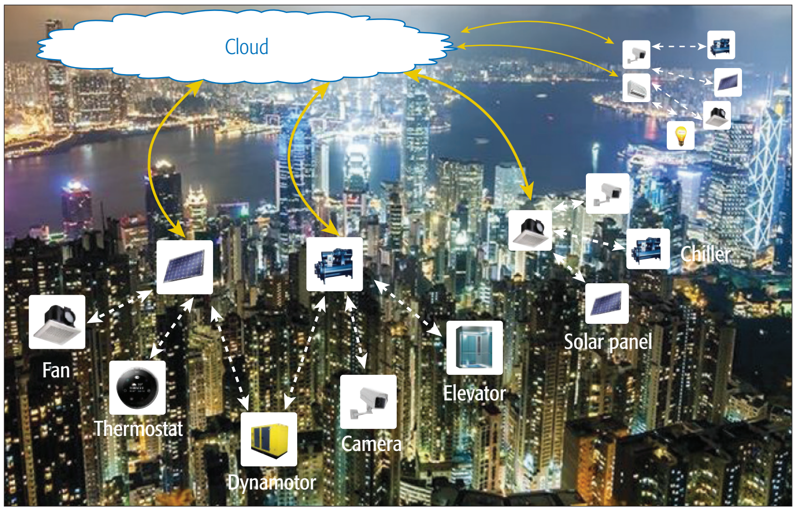

Figure 1. sTube for a smart post-sales application: solid links are thing-to-cloud communication channels; dotted links are local access channels.

Related Architecture

Smart Building Networks: Modern buildings have building automation systems (BASs) to control building equipment [10]. A traditional BAS is mostly signal-based. An sMap architecture [11] was developed to software-define the signal-oriented BAS. In sMap, the IoT devices are organized into a mesh network, and a gateway is used to connect to a BAS control base station. The target of sMap and BAS is to manage thousands of devices from different vendors within a building. The target of sTube is to transmit the data of thousands of IoT devices from the same vendor, spread over hundreds of buildings, to the cloud. sTube differs from sMap in the supporting application *context*. The spread of the devices in buildings controlled by different building owners made the gateway approach infeasible since building-by-building deployment or agreement is needed.

Mobile Phones as Relays: Another recent proposal to transmit IoT data to the cloud is to use mobile phones as relays [12]. The objective is to remove the gateway, which restricts the flexibility. An opportunistic network is constructed where IoT devices will search for nearby mobile phones to relay data. sTube does not rely on opportunistic data transmissions. sTube is clear in *who* should run the transmission function. The IoT devices are from an identified vendor, and the objective is to share the costs of communications and hardware.

Cellular Network/Edge Routers: Multiplexing data flow of different devices is not new. Cellular base stations and edge routers, for example, aggregate data flows. sTube differs from them in *where* to multiplex. The location of the multiplexing function of sTube is on the IoT devices. Traffic flow multiplexing by base stations or the edge routers is controlled by ISPs, but in sTube, it is controlled by the vendors.

We further comment on two foundational networking paradigms, wireless sensor networks (WSNs) [13] and fog computing [14]. In WSNs, since wireless sensors are energy constrained and communication dominates energy consumption, the optimization objective is on all communication links within the WSN. The constraint of

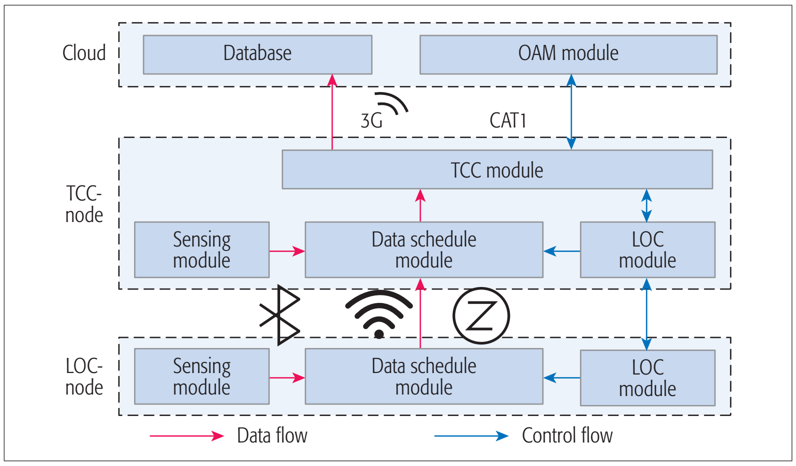

Figure 2. sTube design framework.

sTube is the TCC links between things and clouds. Thus, sTube differs from WSNs in the *optimization objective*. The idea of fog computing is to relocate functions to the edge, for either fast response or cost saving. Fog computing is a conceptual framework. sTube is developed for concrete application scenarios and can be regarded as one instance of fog computing.

sTube Design

The Problem

The overall problem is the mismatch between the limited number of TCC channels and significantly more diversified IoT requirements, but there is currently *no* architecture or system that can support flexibly matching the TCC channels and diverse IoT devices. We call this the *TCC sharing architecture (TCCS-arch)* problem. We would also like to narrow this article by solving the TCCS-arch problem in the scope of a small network. The first-generation SPS applications are usually single-vendor-based, with a small number of nodes, and the data flow patterns are fixed since the sensing data to be monitored are predefined. In this context, we choose a more centralized design approach. We plan to study the TCCS-arch problem in the scope of a large network that has to decentralize control in our future work.

In this article, we address the TCCS-arch problem by developing sTube, a modular architecture for IoT communication sharing, and implement a prototype system.

A Module Design of sTube

There are two types of links in an sTube network: the *TCC links*, which directly connect to the cloud, and *local (LOC) links*, which are local and free. There are two types of nodes in an sTube network: the *TCC-nodes*, which directly connect to the cloud via TCC links, and the *LOC-nodes*, which connect to TCC-nodes via LOC links. The framework is shown in Fig. 2 with three parties: the cloud, the TCC-node, and the LOC-node.

The Cloud: The key module for the cloud is the network operations, administration, and management (OAM) module. It computes the network topology, that is, the peering between LOC-nodes and TCC-nodes; and the data flow transmission schedule, that is, the schedule for the TCC-nodes and LOC-nodes to transmit their data. More specifically, given a *pricing model* in renting the TCC links, this module computes the network topology and data flow transmission schedules that can minimize the cost for the vendor.

We develop a two-phase heuristic algorithm to solve this cost minimization problem. Our algorithm includes an LOC-node peering phase and a data scheduling phase. In the LOC-node peering phase, the algorithm determines to which TCC-node each LOC-node should connect. A simple greedy approach is employed: one by one, each LOC-node selects the reachable TCC-node leading to the minimum incremental cost. In the data scheduling phase, the algorithm determines when to transmit the data from LOC-nodes and the TCC-node sharing the same TCC link. SPS data uploaded at one TCC link are scheduled via the weighted round-robin approach. The network OAM module broadcasts the outcomes of the algorithm to the TCC-nodes and LOC-nodes.

Again, we admit that this is centralized design, but we believe it can serve the initial phase of the first generation SPS applications.

The TCC-Node and LOC-Node: There are three common modules. The sensing module transforms the sensing data into SPS data. The LOC module maintains the link-level connections among the LOC-nodes and TCC-nodes. There are many developed communication technologies for connections among LOC-nodes and TCC-nodes. As shown in Fig. 2, for the TCC links, we can use CAT1, CAT0, or other standards specified by the Third Generation Partnership Project (3GPP); for the LOC links, we can employ WiFi, Bluetooth, Zigbee, and so on. As our primary objective in this article is the sharing of TCC links, We do not limit our design to specific communication technologies to allow flexibility. The data schedule module will store the data and prioritize the transmission of data to the next hop according to the instructions from the network OAM module in the cloud. In our design, the network OAM module will instruct the transmission schedule of the data of LOC-nodes and TCC-nodes. This design can allow a TCC-node to relay the data of the LOC-nodes connecting to it by simply forwarding the packet, without further scheduling at the TCC-nodes.

Reliability and Security Considerations

In production, TCC nodes and LOC-nodes, acting as the communication modules, will be integrated into the product equipment. The failure rate of such communication modules of a device is typically very low. Taking the mobile phone as an example, batteries, screens, and so on are much easier to fail than Bluetooth, WiFi, and 3G/4G modules. Nevertheless, the TCC-node may influence many LOC-nodes. We can over-deploy TCC-nodes so that alternative choices exist when a TCC-node fails. In our implementation, LOC-nodes periodically check the aliveness of the TCC-node.

Regular security problems and solutions of an IoT platform can be found in [15]. A specific concern for SPS is that a vendor does not want its data to be captured by other vendors, that is, it wants to protect the confidentiality and integrity of message transfer and verify the authentication of LOC-nodes, TCC-nodes, and the cloud. We employ asymmetric cryptography using pairs of public and private keys. The current requirement of SPS is single-company-based, so the private keys of TCC-nodes and LOC-nodes can be safely distributed when they are produced.

First, each TCC-node and LOC-node is allocated a burned-in private key when they are produced. The private keys are executable but not readable at the TCC-node and LOC-node devices, and securely cached at the manufacturer's cloud so that an LOC-node or a TCC-node can safely verify and connect to the cloud. The TCC-node and the LOC-node also need to connect to each other through exchanging their public keys. The public keys will be further certified by a manufacturer's private key. In order to achieve this, the manufacturer's public key is given to the LOC-nodes and TCC-nodes when they are produced.

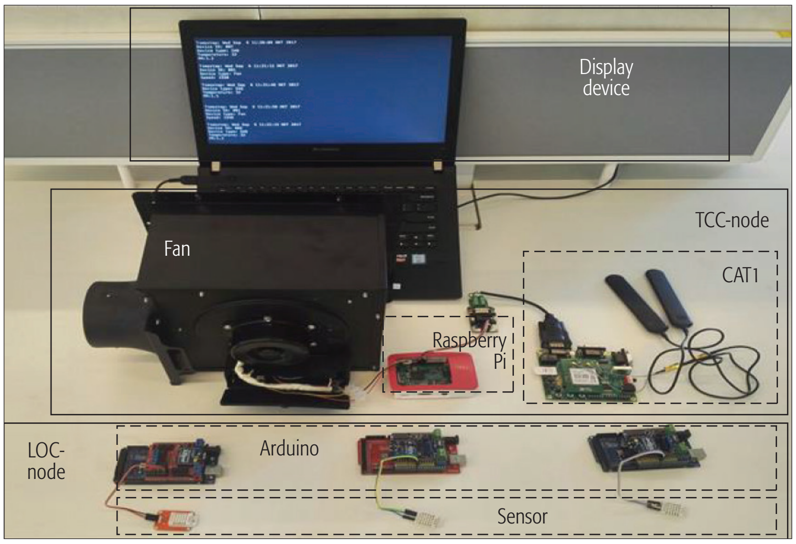

Figure 3. An SPS prototype implementation.

A Case Study with Performance Evaluation

In this section, we present an SPS application with sTube as its underlying architecture support and conduct performance evaluation of this sTube system. We start with the implementation of this SPS application and then present the experiment and simulation results of the performance.

Implementation

We implement sTube and develop an SPS application prototype shown in Fig. 3. The sTube system comprises a laptop computer that displays the cloud end, two sets of TCC-nodes (only one set is shown in the figure since the other set is the same and is out of the scope of the photo), and three sets of LOC-nodes. For the SPS application, we deploy a real fan, a standard component of HVAC systems and indoor air quality (IAQ) sensors.

For the LOC-nodes, Arduino MEGA 2560 is employed as the hardware board and connects to IAQ sensors. For the TCC-nodes, Raspberry Pi 3 Model B is adopted as the hardware board and is connected to the fan. For the TCC-nodes and LOC-nodes, we employ Zigbee to operate LOC communication, realized by XBee S1 on the Arduino and Raspberry Pi, respectively. Our implementation of the TCC module is on a CAT1 board. For the cloud, we rent a server in Ali cloud with 8 cores of 2.5 GHz and a total memory of 128 GB. The data in the cloud are stored in XML format. The announcement of scheduling/pairing from the cloud to LOC nodes is through simple broadcasting. A laptop computer displays the data transmitted to the cloud. 6LowPan is adopted as the packet format throughout the system.

Experiment

System Setup: We consider a network topology with one manufacturer cloud, two TCC-nodes (denoted as T1 and T2), and three LOC-nodes. Each TCC-node may purchase a dedicated CAT1 channel to the cloud. Each LOC-node may establish connection to either one of the two TCC-nodes via a free channel (operating on unlicensed frequency band), so the TCC-node will forward the SPS data from the LOC-node to the cloud. Each LOC-node can only connect to one TCC-node at one time. LOC-nodes cannot communicate with each other. T1, T2, and each LOC-node collect SPS data at rates of 400, 800, and 200 bits every 10 s respectively. Through referencing the price quotes provided by China Telecom, we consider two pricing models in this article: the *pay-as-you-go (PAYG)* model and the *monthly plan (MP)* model.[1] In the PAYG model, at each reserved CAT1 channel every month, \$1 is charged for every 40 MB data usage. In the MP model, at each reserved CAT1 channel every month, \$3 is charged for the first 100 MB data usage, and \$1 is charged for every additional 50 MB data usage.

We evaluate the monetary cost of three schemes:

- *Exclusive channel occupation (ECO)*: Each device transmits its data directly to the cloud via its dedicated purchased CAT1 channel.
- sTube without scheduling (*sTube-NS*) approach: Each LOC-node independently randomly chooses one reachable TCC-node.
- sTube with scheduling (*sTube-S*) approach: The optimal TCC-node-LOC-node association is achieved through the coordination of the network OAM module at the cloud.

We compare the overall monthly costs and delay of the aforementioned three approaches.

Experiment Results: In our experiment, the system is turned on for 6 hours, and the overall data usage is scaled to one month. Then we derive the overall monthly cost of different schemes under our system setting.

We show the results in Fig. 4a by considering the two pricing models, PAYG and MP. We see that the cost of ECO is significantly greater than those of sTube. Under the PAYG model, the overall monthly cost of ECO is \$5. With our sTube design, the cost drops to \$3 for sTube-NS and \$2 for sTube-S: a savings of 40 and 60 percent, respectively. Please note that \$1 per month is affordable to maintain a TCC-node device, such as the fan used in our experiment (a typical AMX brand fan costs \$106.6), but it is expensive for an LOC-node device, such as the IAQ sensor used

[1] For the PAYG model, at each TCC-node, a certain charge is applied when each TCC-node uses up every *L* data volume. For the MP model, the ISPs provide a set of monthly data plans. Each TCC-node can select one data plan at the beginning of each billing cycle.

The TCC-node and the LOC-node also need to connect to each other through exchanging their public keys. The public keys will be further certified by a manufacturer's private key. In order to achieve so, the the manufacturer's public key is given to the LOC-nodes and TCC-nodes when they are produced.

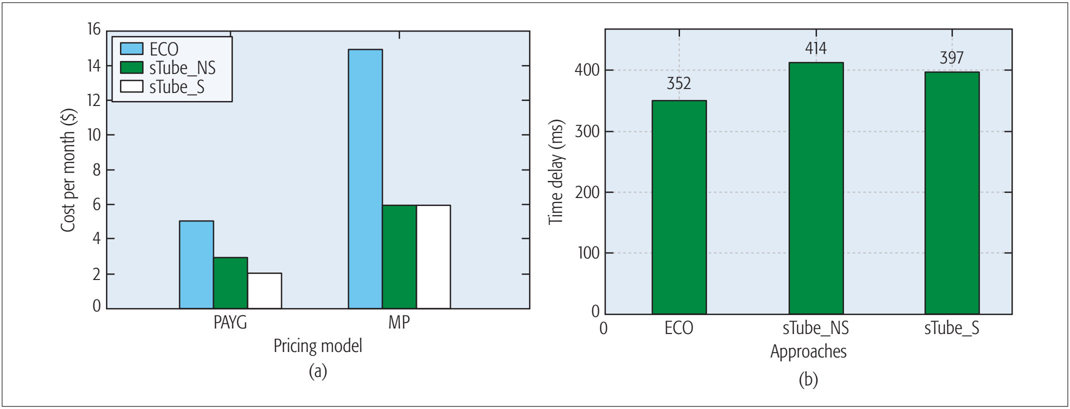

Figure 4. a) Cost per month of three methods under different pricing models; b) time delay under three methods.

in our experiment (an IAQ sensor itself is only $1.35). If ECO is applied, each fan or IAQ spends $1 per month — expensive for IAQs. In contrast, if sTube-S is adopted, we can reasonably let each TCC-node and LOC-node contribute $0.7 and $0.2 per month, respectively, affordable to everyone.

We also compare ECO, sTube-NS, and sTube-S under the MP pricing model. We see that the gap in monthly costs between ECO and sTube becomes even bigger. Clearly, the cost for ECO, $15, is significantly greater than those of sTube-S and sTube-NS, $6, leading to a saving of 60 percent.

We also test the delay performance of ECO, sTube-NS, and sTube-S, that is, the time duration from sending the data packet from IoT to receiving the packet at the cloud. As shown in Fig. 4b, the delay performance of ECO, sTube-NS, and sTube-S is 352 ms, 414 ms, and 397 ms, respectively. However, we want to point out that the SPS application is time-insensitive in many situations. This is because the collected data for the SPS is mainly used for product maintenance and learning the usage patterns, which do not require real-time data and can tolerate hours or even days of delay. Therefore, the small delay introduced by sTube is negligible in this use case.

Simulation

We further verify the performance of sTube via simulation when the number of nodes in the system becomes large.

Simulation Setup: The locations for TCC-nodes and *N*-nodes are randomly and uniformly distributed in a 100 × 100 m^2 square. There are *N* LOC-nodes and 60 TCC-nodes. LOC-nodes can connect to the TCC-nodes in the range of 10 m. The communication range of the TCC-nodes can cover the square. We study the pricing models PAYG and MP employed in the experiment. The data usage of a node is uniformly distributed from 10 MB to 20 MB. We compare ECO, sTube-NS, and sTube-S when the number of LOC-nodes in the system becomes large.

Simulation Results: In Fig. 5a, we show the monthly cost of ECO, sTube-NS, and sTube-S under the two aforementioned pricing models through gradually increasing the number of LOC-node devices.

As we can see from Fig. 5a, the monthly cost of ECO grows significantly as the device number increases. Specifically, an additional $1 is charged for each additional LOC-node device. However, the monthly costs of sTube-NS and sTube-S increase much more slowly. For example, when the LOC-node number is 600, the cost of ECO, $660, is 2.46 times that of sTube-NS and 2.81 times that of sTube-S. When the node number increases from 0 to 400, only an additional $138 for sTube-NS or $114 for sTube-S is charged. This confirms that sTube-NS and sTube-S significantly outperform ECO.

The monthly cost gap between ECO and sTube is even greater under the MP pricing model, shown in Fig. 5b. Only $37 for sTube-NS and $25 for sTube-S are added when the LOC-node device number increases from 0 to 400. However, the monthly cost of ECO increases much faster. When the number of devices is 600, the cost of ECO, $1980, is 7.27 times to that of sTube-NS and 8.01 times that of sTube-NS. This confirms that using sTube can obtain considerable money saving. Moreover, it should be noted that a larger number of devices in the system leads to higher channel sharing opportunities introduced by sTube, and thus higher performance gain brought by sTube.

We also observe that sTube-S further reduces the cost by up to 12 percent when the number of LOC-nodes is 700 under the PAYG model. This suggests that the cloud optimization operation will bring considerable cost saving as well.

Conclusion

In this article, we propose sTube, an architecture for IoT communication sharing, to support applications transmitting the data of IoT devices to the cloud in a self-contained and cost-efficient way. Intrinsically, sTube addresses the mismatch between the limited number of choices of thing-to-cloud communication channels and the large number of diverse communication requirements from the IoT applications on data rates and costs. The evaluation results showed that sTube can save the cost by an order as compared to using dedicated channels for each IoT device. There can be many future research directions for this study. Most importantly, the current design of sTube relies on centralized control in the cloud. This is only suitable for a small network. We plan to develop a decentralized sTube network in the future.

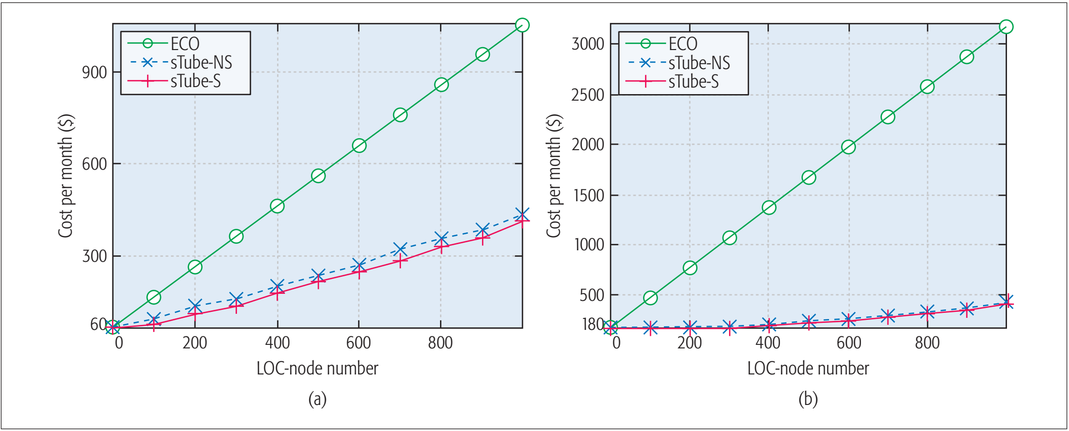

Figure 5. Cost per month of three methods under two pricing models with different numbers of nodes: a) PAYG; b) MP.

The evaluation results showed that sTube can save the cost for an order as compared to using dedicated channels for each IoT device. There can be many future research directions for this study. Most importantly, the current design of sTube relies on a centralized control in the cloud. This is only suitable for a small network. We plan to develop a decentralized sTube network in the future.

References

[1] D. Niyato *et al.*, "Economics of Internet of Things: An Information Market Approach," *IEEE Wireless Commun.*, vol. 23, no. 4, Aug. 2016, pp. 136–45.
[2] A. M. Vegni *et al.*, "A Bayesian Packet Sharing Approach for Noisy IoT Scenarios," *Proc. IEEE IoTDI '16*, Berlin, Germany, Apr. 2016, pp. 305–08.
[3] Z. Zheng *et al.*, "Urban Traffic Prediction Through the Second Use of Inexpensive Big Data From Buildings," *Proc. ACM CIKM '16*, Indianapolis, IN, Oct. 2016, pp. 1363–72.
[4] J. Lee, C. Jin, and Z. Liu, "Predictive Big Data Analytics and Cyber Physical Systems for TES Systems," *Advances in Through-Life Engineering Services*, Springer, 2017, pp. 97–112.
[5] A. Díaz-Zayas *et al.*, "3GPP Standards to Deliver LTE Connectivity for IoT," *Proc. IEEE IoTDI '16*, Berlin, Germany, Apr. 2016, pp. 283–88.
[6] G. Naddafzadeh-Shirazi *et al.*, "Coverage Enhancement Techniques for Machine-to-Machine Communications over LTE," *IEEE Commun. Mag.*, vol. 53, no. 7, 2015, pp. 192–200.
[7] J. Roessler, "LTE-Advanced (3GPP Rel. 12) Technology Introduction," Apr. 2015, White Paper; https://cdn.rohde-schwarz.com/pws/dl downloads/dl application/ application notes/1ma252/1MA252 2e LTE Rel12 technology.pdf, accessed Oct. 2016.
[8] J. Jiang and Y. Qian, "Distributed Communication Architecture for Smart Grid Applications," *IEEE Commun. Mag.*, vol. 54, no. 12, Dec. 2016, pp. 60–67.
[9] S. Latre *et al.*, "City of Things: An Integrated and Multi-Technology Testbed for IoT Smart City Experiments," *Proc. IEEE ISC2 '16*, Trento, Italy, Sept. 2016, pp. 1–8.
[10] J. Gao, J. Ploennigs, and M. Berges, "A Data-Driven Meta-Data Inference Framework for Building Automation Systems," *Proc. ACM Buildsys '15*, Seoul, South Korea, Nov. 2015, pp. 23–32.
[11] S. Dawson-Haggerty *et al.*, "sMAP: A Simple Measurement and Actuation Profile for Physical Information," *Proc. ACM SenSys '10*, Zurich, Switzerland, Nov. 2010, pp. 197–210.
[12] T. Zachariah *et al.*, "The Internet of Things Has a Gateway Problem," *Proc. ACM HotMobile '15*, Santa Fe, NM, Feb. 2015, pp. 27–32.
[13] I. F. Akyildiz *et al.*, "Wireless Sensor Networks: A Survey," *Computer Networks*, vol. 38, no. 4, 2002, pp. 393–422.
[14] M. Chiang and T. Zhang, "Fog and IoT: An Overview of Research Opportunities," *IEEE Internet of Things J.*, vol. 3, no. 6, 2016, pp. 854–64.
[15] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A Survey," *Computer Networks*, vol. 54, no. 15, 2010, pp. 2787–2805.

Biographies

Dan Wang [S'05, M'07, SM] received his B.Sc degree from Peking University, Beijing, China, in 2000, his M.Sc degree from Case Western Reserve University, Cleveland, Ohio, in 2004, and his Ph.D. degree from Simon Fraser University, Burnaby, British Columbia, Canada, in 2007, all in computer science. He is currently an associate professor at the Department of Computing, Hong Kong Polytechnic University. His research interests include Internet architecture, QoS, economics, and smart buildings.

Wei Bao received his Ph.D. degree in electrical and computer engineering from the University of Toronto, Canada, in 2016. He is currently a lecturer at the School of Information Technologies, University of Sydney, Australia. His research covers the area of network science, with particular emphasis on 5G systems, the Internet of Things, and mobile computing.

Chuang Hu received his B.Sc degree and M.Sc degree from the Department of Computing Science, Wuhan University, China, in 2013 and 2016, respectively. He is currently a Ph.D. student in the Department of Computing, Hong Kong Polytechnic University. His research interests include networking, wireless technologies, IoT, cloud computing, and big data.

Yi Qian (yqian2@unl.edu) is a professor in the Department of Electrical and Computer Engineering, University of Nebraska-Lincoln (UNL). Prior to joining UNL, he worked in the telecommunications industry, academia, and the government. His research interests include information assurance and network security, network design, network modeling, simulations and performance analysis for next generation wireless networks, wireless ad hoc and sensor networks, vehicular networks, smart grid communication networks, broadband satellite networks, optical networks, high-speed networks, and the Internet.

Muqiao Zheng (joe.zheng@fusquare.con) is a student in the Department of Automation, Guangdong University of Technology. He is interested in IoT, wireless technologies, and embedded software development.

Shi Wang is pursuing an M.Sc. in information technology at Hong Kong University of Science and Technology. She is interested in the Internet of Things, cognitive technologies, big data, and software development.