sTube+: An IoT Communication Sharing Architecture for Smart After-sales Maintenance in Buildings

Chuang Hu
Hong Kong Polytechnic University

Wei Bao
Sydney University

Dan Wang
Hong Kong Polytechnic University

Yi Qian
Nebraska Lincoln University

Muqiao Zheng
Guangdong Technology University

Shi Wang
Hong Kong Polytechnic University

ABSTRACT

Nowadays, manufacturers want to send the data of their products to the cloud, so that they can conduct analysis and improve their operation, maintenance and services. Manufacturers are looking for a self-contained solution. This is because their products are deployed in a large number of different buildings, and it is neither feasible for a vendor to negotiate with each building to use the building's network (e.g., WiFi) nor practical to establish its own network infrastructure. The vendor can rent a dedicated channel from an ISP to act as a thing-to-cloud communication (TCC) link for each of its IoT devices. The readily available choices, e.g., 3G is over costly for most IoT devices. ISPs are developing cheaper choices for TCC links, yet we expect that the number of choices for TCC links will be small as compared to hundreds or thousands of requirements on different costs and data rates from IoT applications.

We address this issue by proposing a *communication sharing* architecture *sTube+, sharing tube*. The objective of sTube+ is to organize a greater number of IoT devices, with heterogeneous data communication and cost requirements to efficiently share fewer choices of TCC links, and transmit their data to the cloud. We take a design of centralized price optimization and distributed network control. More specifically, we architect a layered architecture for data delivery, develop algorithms to optimize the overall monetary cost, and prototype a fully functioning system of sTube+. We evaluate sTube+ by both experiments and simulations. In addition, we develop a case study on smart maintenance of chillers and pumps, using sTube+ as the underlying network architecture.

KEYWORDS

IoT, communication architecture, smart building, thing-to-cloud

1 INTRODUCTION

One important value proposition of the Internet of Things (IoT) is the data generated by the IoT devices (a.k.a, 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 adding value can be significant [2]. For example, it has been shown that big building data (e.g., carbon dioxide (CO2) data from the heating, ventilation and air conditioning (HVAC) systems) can be exploited to predict traffic status of nearby roads [3]. Smart After-sales Maintenance and Services (SAMS), which will become the case study of this paper, is another example. Manufacturers of air conditioners, pumps, elevators, etc., are now transforming their machinery into smart machinery. When sending the data of their products to the cloud, SAMS can operate in a trouble-preventing mode instead of trouble-shooting mode. This can substantially improve the quality and reduce the cost of the product maintenance. Moreover, manufacturers can learn the usage patterns of their customers. Thus they can recommend other products and develop top-up services based on such knowledge [4].

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 vendor may develop a WiFi network for the IoT application. However, WiFi needs additional infrastructure, e.g., a gateway that finally relays data to the cloud. This is not suitable for SAMS. For example, a vendor would like to monitor all its air conditioners in a region, installed in a large number of buildings. The WiFi choice needs deployment of WiFi networks on a building-by-building basis. In other words, the vendor is developing a separated network infrastructure. If using existing WiFi networks in the buildings, there will be policy and security concerns. A building can easily have products from tens of vendors. If each vendor wants its equipment to infiltrate the WiFi network of the building, building operators need to bear overwhelming liability. Simply-put, applications such as SAMS are looking for an infrastructure-less solution.

The vendor may rely on the infrastructure of a service provider (ISP) and rent a dedicated wireless communication channel for each IoT device [5] to support the *thing-to-cloud communication (TCC)* links. Current choices for TCC links are very limited. The readily available 3G/4G is 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, yet we will face hundreds, if not thousands, of heterogeneous requirements. In the SAMS example, the cost of CAT1 might be justifiable for a chiller, yet it may be too costly for a fan.

We see a clear gap between the possible choices of TCC links, and the number of requirements on different costs and data rates

BuildSys '17, November 8–9, 2017, Delft, Netherlands

ACM ISBN 978-1-4503-5544-5/17/11…$15.00
https://doi.org/10.1145/3137133.3137143

from the IoT applications. To address this issue, we propose Sharing Tube plus (sTube+) 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 fewer choices of TCC links, and transmit their data to the cloud. An example SAMS application using sTube+ is shown in Fig. 1.

To bring sTube+ into reality, the challenges not only lie in the TCC link sharing optimization, but also that there is currently *no* architecture for IoT communication sharing data delivery. We propose a design approach of centralized price optimization and distributed network control. We architect a layered architecture for data delivery, optimize TCC link sharing, and prototype a functioning sTube+ system. We evaluate sTube+ with experiments and simulations. Finally, we present a SAMS case study. In this case study, we collect data from chillers and pumps, two core components of a centralized HVAC system, and analyze their performance in the cloud. sTube+ serves as the underlying architecture in this case study.

The contributions of the paper can be summarized as:

- To the best of our knowledge, we are the first to clarify the necessity, scope and example applications of IoT communication sharing, and we discuss why existing architectures cannot meet the requirement (Section 2).
- We design a layered architecture for IoT communication sharing data delivery (Section 4). We formalize a set of problems for TCC link sharing optimization, and develop algorithms with provable bounds (Section 5). We prototype a fully functioning system for sTube+ (Section 6).
- We comprehensively evaluate sTube+ (Section 7). In particular, we develop a SAMS case study, using sTube+ as the underlying architecture (Section 8).

2 THE MOTIVATION AND RELATED ARCHITECTURE

To ensure that a network architecture to be practically useful, it is necessary to clarify its application scenarios and scope. We believe that SAMS will be one killer application for sTube+. Since SAMS is still at an emerging stage, we first briefly analyze an example SAMS and its benefits. We then analyze the scope of sTube+, i.e., when sharing is a must or superior. Finally, we discuss the differences between existing architectures and sTube+.

2.1 Chiller Maintenance: How SAMS Benefits

We are currently working on a real SAMS on centralized HVAC systems. We analyze the benefit of SAMS by using chillers, one core component of an HVAC system, as an example.

The current chiller maintenance consists of routine maintenance and emergency repair, and their respective costs are USD $897.12 and USD $5639.94 (we use USD as the monetary unit in the rest of this paper) per time [8]. An optimal maintenance plan is a balance of routine maintenance and emergency repair. This is usually done by analyzing the degradation of chillers. Intuitively, routine maintenance will be more frequent if a certain type of chiller degrades faster. Chiller degradation is affected by many factors, such as its intrinsic reliability and the usage pattern of the chiller. Note that though the chiller reliability can be extensively tested in labs, the usage pattern of a chiller is determined by customers, and is difficult to know at the time that this chiller is being manufactured. This is one key reason that SAMS can become superior.

The key indicator for the performance (degradation) of a chiller is Coefficient of Performance (COP) [9]. Maintenance is needed if the COP of a chiller is below a certain threshold.[1]

To compare the current maintenance plan and SAMS, we obtained four year data of ten chillers in three buildings. We calculated the optimal plan for current maintenance with a routine maintenance interval of 3.1 months (detailed calculation in [10]), leading to a cost of $4052.64 per chiller per year. For SAMS, we can collect the chiller data in real time. The cost reduces to $2813.66, with an average maintenance interval of 3.89 months. This leads to a 30.58% saving. Note that this is only a baseline comparison. If we consider joint maintenance of multiple equipment, a prediction of equipment degradation, and that current maintenance plan has to be conservative (e.g., shorter than 3.1 months), we can expect a much greater gain from SAMS.

2.2 Communication Channels: Why Do 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. Yet 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 cannot find a well-matched thing-to-cloud communication channel.

Readily available self-contained solutions, e.g., 3G/4G [11], are provided by ISPs. 3G/4G are over powerful and expensive for most IoT applications. Alternative solutions include LTE Category 1 (CAT1) released in 2016 and the to-appear LTE Category 0 (CAT0). New choices are being developed, yet the progress can not match the surging requirements. More importantly, there may be requirements that will never be developed by ISPs. For example, CAT1 has a monthly cost at around $1 for a data volume of 45 MB. Assume that an equipment has a data volume of 50 MB but it can only afford $1. ISPs will not deliberately develop such plan since it makes CAT1 non-marketable. In a sharing environment, a close-by 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, etc. They are good for device-to-device communication. WiFi, LoRa and SigFox [12] 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 paper, however, will not emphasize on the design of the LOC networks.

2.3 Related Architecture to sTube+

Smart Building Networks: Modern buildings have building automation systems (BAS) to control building equipment [13]. Traditional BAS is mostly signal-based. An sMap architecture [14] was developed to software-define traditional BAS. In sMap, the IoT devices are organized into a mesh network, and a gateway is used.

[1] A low COP does not mean a direct chiller failure; yet it indicates sensible human comfort down grade and substantial energy usage inefficiency. The current threshold imposed in Country/City Hong Kong is 5.7.

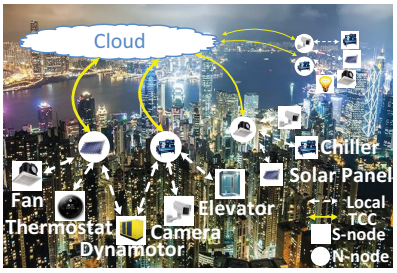

Figure 1: Smart After-Sales Maintenance Services (SAMS).

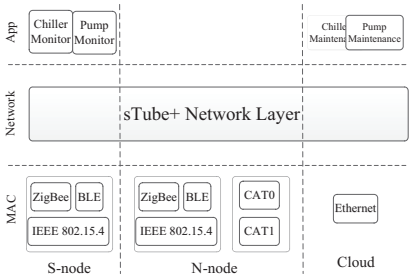

Figure 2: A Layered Architecture.

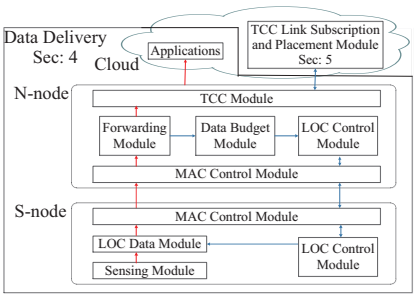

Figure 3: sTube+ module design.

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, of the same vendor, spread at 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 a building-by-building based deployment or agreement is needed.

Mobile Phones as Relays: One recent proposal to transmit IoT data to the cloud is to use mobile phones as relays [15]. The objective is to remove the gateway, which restricts the scalability. 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+ differs as it is clear on *who* should run the transmission function.

Cellular Network/Edge Routers: Multiplexing data flow of different devices is not new. Cellular base stations and edge routers 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/edge routers is controlled by ISPs; yet in sTube+, it is controlled by the vendors.

3G Data Sharing: Data sharing is not new. One example is the hotspot function of mobile phones. 3G hotspot is local to a few phones, and a simple master and slave design is enough. The requirements for sTube+, as represented by SAMS applications, need a scalable architecture that can handle the heterogeneity of the hardware devices, multi-path routing, an overall optimization of the cost of a vendor, etc. The level of *complexity* differs greatly. Another example is represented by family plans, where multiple sim cards are allowed. Yet family plans cannot share *different plans*, and in our scenario, the vendor may register different plans for overall optimization. In addition, it is questionable whether ISPs will provide plans where thousands of sim cards, in particular those with a large amount of small-size flows, can share a single plan. Intrinsically, this means that ISPs take the burden and cut their own profits for the benefit of vendors. As such, we believe that ISPs will impose certain *limit* even if plans with multiple sim cards are developed; making the vendor side sharing still important.

We further comment on two foundational networking paradigm **Wireless Sensor Networks** (WSN) [16] and **Fog Computing** [17]. In WSN, 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 sTube+ is the TCC links between things and clouds. Thus, sTube+ differs from WSN in the *optimization objective*. The idea of Fog Computing is to relocate functions to the edge, either for a fast response or for 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.

3 THE PROBLEM AND DESIGN OVERVIEW

In sTube+, there are two types of links, the thing-to-cloud communication (TCC) links that directly connect to the cloud, and local (LOC) links that are local and free. There are three types of nodes (see Fig. 1), sensing nodes (connected to the SAMS equipment), nodes with TCC links, and the cloud servers. In this paper, we call them S-nodes, N-nodes and the clouds. Note that S-node and N-node can be installed on the same physical equipment.

The problem is that given a pricing model of the TCC links, a set of data volume requirements of the S-node, and the possible locations for S-nodes and N-nodes, develop a TCC link/N-node subscription and placement scheme, as well as a scheme for data delivery between S-nodes and the cloud so that the overall monetary cost of the TCC links can be minimized.[2]

The challenge is that there is currently *no* architecture for data delivery; yet the optimization for TCC link subscription is affected by the data delivery architecture. For example, a fully centralized architecture may lead to a joint optimization of TCC link subscription, placement and the routing between S-nodes and the cloud.

We first clarify the features of the architecture: 1) the cloud has the knowledge of all S-nodes, e.g., the vendor should know all its equipment; and 2) S-nodes have heterogeneous requirements in data volume, incremental deployment, new future functions, etc.

To this end, we choose a design of *centralized price optimization* and *distributed network control*. More specifically, since the cloud has the knowledge of the location and the rough data rates of all S-nodes, it can compute, e.g., monthly, an overall optimization of the TCC link subscription and placement. Yet for packet delivery, and micro-level topology dynamics such as the peering of N-nodes and S-nodes, a distributed network control is needed for scalability.

We first design a layered architecture that supports data delivery of the S-nodes to the clouds (Section 4). We then formulate the TCC link sharing problem and develop algorithms. Note that the TCC link sharing optimization is a separate module from the data delivery architecture (Section 5). In addition to cost optimization, sTube+ needs to be reliable itself. Otherwise, we will be maintaining sTube+ rather than the equipment. We achieve this by over deployment of the TCC links (Section 5.4), as well as topology and

[2]We assume that the communication cost dominates because it is a monthly recurrent cost. We ignore the hardware cost in this paper.

data delivery recovery when an N-node fails (Section 4.1). We also discuss a special security concern where a vendor does not want its SAMS data to be captured by other vendors (Section 4.3).

4 THE STUBE+ ARCHITECTURE

4.1 A Layered Architecture for Data Delivery

4.1.1 An End-to-End Approach. In SAMS, each S-node represents an equipment. Even though in our scenario, all equipment belongs to the same vendor, they differ greatly in operation, maintenance and services. Each of the S-node and its associated cloud application can be individually developed, e.g., by sub-divisions of the vendor, and there may need possible future function extensions. We thus choose an end-to-end approach and let N-nodes only be responsible for traffic forwarding. From the application's point of view, the S-node talks with the cloud directly, see Fig. 2.

Note that the end-to-end approach requires the hardware of the S-nodes to be able to support the IP layer. We believe that this is reasonable since the accumulated value of the collected data in long-term should outweigh such one-time hardware overhead.

4.1.2 Network Topology Control. With an end-to-end design, the cloud, N-nodes and S-nodes are all involved in the network layer. We now study how the topology/nexthops should be managed.

For a very small scale network, the cloud can adopt a centralized design, where it computes all connections and broadcasts the peering results. For a general network, the cloud should not be triggered by micro-level dynamics, i.e., the peering among N-nodes and S-nodes. We choose to let the cloud only manage and monitor the *data budgets* of N-nodes, i.e., the data volume allocated to an N-node for its TCC link in a period of time. Note that the data budgets of the N-nodes in a sub-area may be exhausted because certain S-nodes have unexpected traffic, other N-nodes fail, new S-node joins, etc. Nevertheless, the number of N-nodes is much smaller than the number of equipment in the system and the frequency of budget allocation and updates is low.

The nexthop of an N-node is the cloud directly.[3] In this paper, we also do not consider multiple wireless hops where an S-node uses other S-nodes to relay its data. As such, the remaining issue is to settle the peering between S-nodes and N-nodes.

Our objective is to minimize the complexity. We make two choices. First, we choose to let S-nodes take the initiative to manage the peering with the N-nodes. In particular, an S-node may need to use the data budget of multiple N-nodes. Therefore, letting S-nodes, rather than N-nodes, take the initiative has much fewer overheads. Second, we develop an N-node peering algorithm, where each S-node makes independent decisions, yet the joint force collectively adapts to various network and data budget dynamics.

The N-node peering algorithm (N-peering) of the S-nodes: Each S-node maintains a set of neighboring N-nodes. Each N-node periodically broadcasts its *residual budget-index (rb-index)* to all its neighboring S-nodes. This rb-index is designed as an increasing function of its remaining data budget. Periodically, S-node will select to connect to one N-node based on these rb-indices sent from its neighboring N-nodes. Specifically, let $\mathcal{N}$ be the set of neighboring N-nodes of an S-node and I_i denote N-node i's rb-index. The S-node computes connection probabilities to each neighboring N-node as $p_i = \frac{I_i}{\sum_{j\in\mathcal{N}} I_j}$, and connects to one of them according to these probabilities. As a consequence, the N-node with a greater remaining budget has a greater probability to be selected. We show the nodes interact of the N-node peeing process in Fig. 4.

[3]Theoretically, an N-node can route from other N-nodes; yet this increases the complexity and is not necessary for common cases.

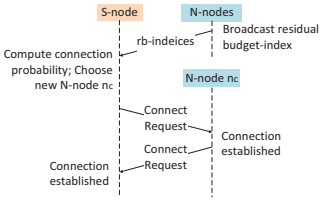

Figure 4: Nodes interaction when periodically choosing N-node.

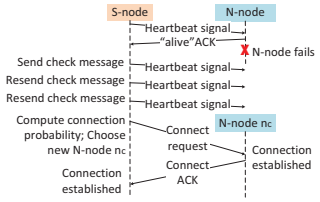

Figure 5: Nodes interaction when the connected N-node is failure.

Neighbor maintenance of the S-nodes: Each S-node periodically sends heartbeat signals to check the availability/failure of its neighboring N-nodes, and updates neighbor if the original neighbor fails. The process is shown in Fig. 5. Specifically, the S-node periodically sends heartbeat signal to the connected N-node and waits "alive" ACK from it. If the S-node does not receive respond from the N-node in time t, it resends heartbeat signal to the connected N-node. If the S-node does not receive any respond after three heart signal, the S-node regards the N-node as failure and updates to a new neighbor. To minimize possible data loss, the S-nodes with a higher data rates will have a shorter checking period. Let T_{ci} be the *checking period* of S-node i. Let D_i be the successive data loss that can be tolerated. Let r_i be the data rate of S-node i. Let T_u be the period needed to connect to a new neighbor. The S-node i sets $T_{ci} = \frac{D_i}{r_i} - 3t - T_u$. Due to the reliable transmission provided by CoAP (described in Section. 6.1), the data loss of an S-nodes occurs only when the connected N-nodes fails.

4.2 Detailed Modules for a Functioning System

A functioning system has modules of all layers, see Fig. 3.

S-nodes have four modules. The *sensing* module connects to the equipment and collects sensing data. The *MAC control* module maintains the data link level connection between itself and the N-nodes within its communication range. The *LOC control* module maintains the network topology. The *LOC data* module transmits the data to the N-node.

N-nodes have five modules. *MAC control* module maintains the data link level connection between itself and the S-nodes. The *forwarding* module relays the data received from its MAC layer by forwarding the packet. The *TCC* module maintains the data link level connection between the N-node and the clouds. The LOC control module answers network layer queries from S-nodes. The *data budget* module maintains its data usage and accepts recharging from the cloud if necessary.

The cloud runs applications. The cloud has a centralized *TCC link subscription and placement* module. It computes data budgets (details in Section 5) of N-nodes.

4.3 Security Concerns

IoT systems face various security problems. Common problems and solutions can be found in [18, 19]. A specific security concern for

SAMS is that a vendor does not want its data captured by other vendors. For example, an attacker may eavesdrop the data transmission from an S-node, or fake the identity of an N-node to conduct a man-in-the-middle attack. Here, the challenge in sTube+ is that S-nodes cannot connect to the Internet directly. As such, we need to maintain the integrity of N-nodes.

We address this problem by a simple authentication design. First, since each N-node is able to connect to the Internet, the communications between an N-node and the cloud can be safely established by using standard Transport Layer Security (TLS) protocols. Second, an S-node and N-node should also be able to verify each other and establish a safe communication link. This can be achieved via exchanging their public keys. The main issue here is that how the S-node and N-node can verify each other's public key when S-node is disconnected from the Internet. In our scenario, since S-nodes and N-nodes are produced by the same manufacturer, the manufacturer can hard code the certificate (derived from the manufacture's private key) when the node is produced, i.e., certificate pinning. The manufacture's public key is also pinned to the node. As a result, the two parties are able to verify each other even if they are disconnected from the Internet.

5 TCC LINK SHARING OPTIMIZATION

The TCC link sharing problem is to answer which N-node (location) should reserve a TCC link from the ISP and how much budget they should reserve so as to minimize the overall monetary cost. We formulate two TCC link sharing problems according to two widely accepted pricing models. We show both problems are NP-complete and propose approximation algorithms. We then study the problem to maintain the reliability of sTube+.

5.1 Problem Formulation and Analysis

5.1.1 Network Topology. Let $\mathcal{N} = \{n_1, n_2, ..., n_N\}$ denote the set of locations of N-nodes. An N-node can be either *installed* or *vacant*. Let $f(n_j) = 1$ if n_j is installed; $f(n_j) = 0$ if n_j is vacant.

Let $\mathcal{S} = \{s_1, s_2, ..., s_M\}$ be the set of S-nodes. S-node s_i's data usage in one billing cycle is u_i. Let $\mathcal{S}_j$ denote the subset of S-nodes, which can reach N-node n_j. The term "reach" means that it is possible for the S-node to deliver its data to the N-node through some LOC links. We assume each S-node can reach at least one N-node.

Let u_{ij} be the amount of data uploaded via n_j. We have $u_i = \sum_{j:f(n_j)=1 \text{ and } s_i \in \mathcal{S}_j} u_{ij}$. If n_j is installed, the load at the TCC link of n_j, denoted by U_j, is the accumulated data amount uploaded by its connected S-nodes. We have $U_j = \sum_{i:s_i \in \mathcal{S}_j} u_{ij}$. Otherwise, $U_j = 0$.

5.1.2 Pricing Model. We consider two widely adopted models, the pay-as-you-go (PAYG) model and the monthly-plan (MP) model.

For the PAYG model, at each N-node, a certain charge is applied when the each N-node uses up every L data volume,i.e., $C_{PAYG}(x) = \sum_{l=1}^{\lceil \frac{x}{L} \rceil} P_l$. In practice, $P_l \geq P_{l+1}, \forall l$ [20], and $\lim_{l \to +\infty} P_l = P_{min}$.

There are two special cases of the PAYG model. The first is that the price for all data volume steps are equal, called PAYG-E. The second is the all-you-can-use (AYCU) model, where the price is b if an N-node is installed and 0 if vacant. The amount of data usage is unlimited under this pricing model.

For the MP model, the ISPs provide a set of monthly data plans, denoted as $\mathcal{D}$. Each N-node can select one data plan from $\mathcal{D}$ at the beginning of each billing cycle. The data plan can not be changed in one billing cycle. For monthly data plan $m_h \in \mathcal{D}$, the price is represented by Eq. (1). Price c_h is charged for a fixed amount of cap usage k_h; If the data usage hits this cap then a higher price d is charged for each per data usage unit.

$$C_{m_h}(x) = \begin{cases} c_h & , x \leq k_h, \\ c_h + d(x - k_h) & , x > k_h. \end{cases} \quad (1)$$

5.1.3 Problem Formulation.

Problem 1 (PAYG TCC Sharing). *Given the locations of S-nodes and N-nodes, the monthly data volume of S-nodes, determine the placement of N-nodes and the amount of data uploaded via N-nodes for each S-node, to minimize the sum PAYG cost over all N-nodes.*

The MP model provides a set of monthly data plans. Choosing different data plans for N-nodes will cause different overall monetary costs. Let $d_j \in \mathcal{D}$ be the data plan employed by N-node n_j.

Problem 2 (MP TCC Sharing). *Given the locations of S-nodes and N-nodes, the monthly data volume of S-nodes, determine the placement of N-nodes, the employed data plan for N-nodes, and the amount of data uploaded via N-nodes for each S-node, to minimize the sum MP cost over all N-nodes.*

5.1.4 Problem Analysis.

Theorem 1. *Problems PAYG TCC Sharing and MP TCC Sharing are both NP-complete.*

Proof. We prove this theorem by transforming both problems into the set cover problem [21]. Due to page limitation, all proofs of this paper can be found in [10]. □

We note that intrinsically, the complexity comes from N-node covering S-nodes (the placement of TCC links/N-nodes), rather than from the pricing model (the subscription of the TCC link prices for the N-nodes).

5.2 The Approximation Algorithm for PAYG TCC Sharing

5.2.1 The Algorithm. The overall problem can be divided into two subproblems: TCC link placement to cover all S-nodes, and subscription of a pricing plan at each placed N-node. The TCC link placement to cover all S-nodes is a set cover problem. We adopt the greedy algorithm in [21]. For price subscription, we develop a simple algorithm where every N-node subscribes P_1, i.e., the 1st step price, and recharge P_l ($l = 2, 3, ...$) if necessary. We call this algorithm Fast N-node Deployment (FND).

We would like to comment that FND will output a placement of N-nodes with an implicit assumption that the S-nodes covered by an N-node will peer with this N-node. Such best peering needs a centralized control (e.g., after the cloud runs FND, it has to inform all N-nodes and S-nodes). In our sTube+ design, the peering control is distributed to S-nodes.

5.2.2 Approximation Ratio Analysis.

Theorem 2. *The approximation ratio of FND is* $\frac{2P_1^2}{P_{min}^2}(\ln M + 1)$.

Proof. The proof of Theorem 2 is non-trivial. There is a series of transformation, which can be found in [10]. □

Please note that the factor $\ln M$ stems from the greedy set cover algorithm [22]. It is the best-known approximation ratio in solving the set cover problem within a polynomial complexity. Since the TCC sharing problem is more complicated than the set cover problem, the factor $\ln M$ is unavoidable in this scenario.

5.3 The Algorithm for MP TCC Sharing

Algorithm 1 N-node placement and subscription, NPS($\mathcal{S}, \mathcal{N}$).

1: Initialize $\mathcal{Z} = \emptyset, N' = 0, minset = \emptyset, mincost = +\infty$
2: $[N', \mathcal{Z}] \leftarrow$ greedySC($\mathcal{S}, \mathcal{N}$)
3: **while** $N' < |\mathcal{N}|$ **do**
4: $[N', \mathcal{Z}] \leftarrow$ Node-Partition($\mathcal{Z}, N'$)
5: $[minset, mincost] \leftarrow$ Binary-Search-Cost($\mathcal{Z}, N'$)
6: **end while**
7: **return** *minset*, *mincost*

For the MP pricing model, the pricing plan does not have a convex structure. We develop an N-node placement and subscription (NPS) algorithm for MP pricing model. This algorithm is divided into two sub-functions, `Node-Partition()`, and `Binary-Search-Cost()`. Given a node number N', `Node-Partition()` will decide a covering scheme by using N' nodes. Given a node covering scheme $\mathcal{Z}$, the sub-function `Binary-Search-Cost()` will search for the minimized cost for this covering sets.

The overall algorithm `NPS()` is an iterative algorithm. It starts from the minimum set cover (line 2, `greedySC()`) and then the sub-function `Node-Partition()` will gradually increase the number of N-nodes (line 4). Then the sub-function `Binary-Search-Cost()` will determine the cost of such partition (line 5). The algorithm stops when the number of nodes is greater than the number of possible locations of N-nodes (line 3).

5.4 Improving the Reliability of sTube+

Both S-node and N-node may fail. We assume that the probability that an S-node fails, p_s, is much less than the equipment itself. In production, S-node (as the communication module) should be integrated into the equipment. We see that the failure rate of the communication module of a device is typically very low. Taking the mobile phone as an example, batteries, screens, etc are much easier to fail than the bluetooth, WiFi, and 3G/4G modules.

We say that an S-node experiences a *service outage* if it cannot reach any N-node or it experiences a failure. Let the probability of failure of N-nodes be p_n. The service outage probability of s_i can be computed as $p_{\text{out}}(s_i) = 1 - (1 - p_s)(1 - p_n^{R_i})$, where R_i is the number of N-nodes that is reachable by s_i. As such, the average outage probability of S-nodes is $p_{\text{out}} = \frac{1}{M}\sum_{i=1}^{M} p_{\text{out}}(s_i)$.

Problem 3 (TCC Sharing-Availability). *Let* p_{req} *be a threshold of the required average outage probability, we aim to deploy enough N-nodes such that* $p_{\text{out}} < p_{\text{req}}$.

We develop algorithm TCC-OD with over-deployment of N-nodes as follows. We compute p_{out} under the current network topology. If $p_{\text{out}} < p_{\text{req}}$, we iteratively install one additional N-node that can maximally decrease p_{out}.

5.5 S-Node Change

If a new S-node is deployed and there is no N-node within its range, a new N-node will be installed for PAYG and MP models. For other arrivals and departures of S-nodes, FND can be rerun for PAYG at anytime, NPS can be rerun at the next billing cycle for MP model.

6 IMPLEMENTATION

We present an implementation of the sTube+ architecture. This includes the MAC layer, network layer, and application layer. We present a case study of a SAMS in Section 8, where we develop the sensing module to collect data from real equipment and conduct data analytics in the cloud.

6.1 The Network Stack

MAC layer: We implement IEEE 802.15.4, ZigBee, and Bluetooth as the MAC layer for the LOC network. We choose CAT1 as the MAC layer for the TCC link.

Network layer: We choose 6LoWPan (IPv6) as the networking layer protocol. There are two special challenges.

The first is that our CAT1 only supports IPv4. Moreover, it only provides application layer interfaces. Thus, we develop an IPv6-IPv4 converter. It locates in the application layer of the N-node (see Fig. 6), yet it emulates the network layer. It has two functions: packet format transformation and IPv6-IPv4 address mapping.

For packet format transformation, the packet we get from the LOC network is an IPv6 packet. We remove all headers to get the application packet. Then we put such packet to the CAT1 interface. The address mapping is done by mapping a group of IPv6 address to an IPv4 address (the address of CAT1) and a port. Every N-node establishes a table of the mapping. Each entry in this table is automatically inserted when the first packet from the S-node reaches the N-node, i.e., N-node allocates each S-node connected to it a universal port with the CAT1's IPv4 address.

The second challenge is that in practice, an S-node should have a fixed IP address. Yet in our implementation, each S-node gets its IPv6 address from N-node using the uIP library from Contiki, making the IP address dynamic. Since the interaction between an S-node and the cloud is bi-directional, the dynamic IP address can break the interaction. To this end, in the application layer, we develop a notification mechanism such that if the IP address of the S-node changes, the S-node will notify the cloud.

Application layer: We use CoAP and UDP for application layer protocols. sTube+ chooses the optional reliable transmission model of CoAP. Specifically, reliable transmission in CoAP is achieved by marking individual messages with the confirmable flag. When the cloud receives a confirmable message, it responds with an acknowledgment message to let the S-node know the message arrived. The

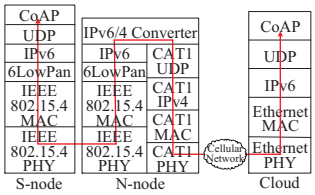

Figure 6: End-to-End Communication.

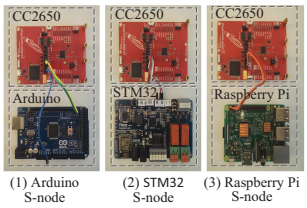

Figure 7: The S-node.

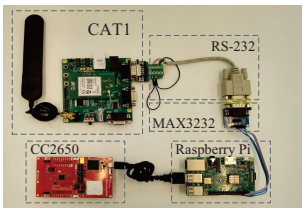

Figure 8: The N-node.

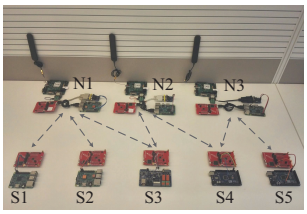

Figure 9: The network topology of the experiments.

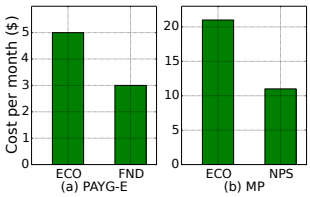

Figure 10: The monthly cost of different schemes.

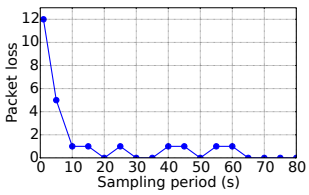

Figure 11: The packet loss as function of sampling period.

S-node will automatically retransmit a confirmable message if an acknowledgment message is not received in the timeout interval.

6.2 Hardware Choices

The S-nodes: We use Arduino MEGA 2560, STM32 and Raspberry Pi 3 Model B as the S-node hardware board (Fig. 7) by considering the different requirements of the capability of hardware board from the equipment and the hardware cost. For example, for the Fan, only the speed of fan should be sensing and the cheap Arduino board can meet its requirement; While for the chiller, the S-node gains the sensing data from the chiller control interface and Modbus RTU protocol should be run on hardware board, thus the more powerful and expensive raspberry pi should be adopted. For the LOC module, we use a Texas Instruments CC2560 SimpleLink^{TM} Wireless MCU for the 802.15.4 radio interface.

The N-nodes: We use a Raspberry Pi 3 Model B as the N-node platform (Fig. 8). For LOC side, we use a Texas Instruments CC 2560 SimpleLink^{TM} Wireless MCU for the 802.15.4 radio interface. Then this module is connected to Raspberry Pi using a USB-to-serial cable. For the TCC side, As the interface of Raspberry Pi is TTL, while the interface provided by CAT1 is RS-232, we use the MAX3232 as a converter. The baud rate of the serial port is 19200 bits per second, i.e. 2400 bytes per second. The CAT1 module supports speeds of 5 Mbps upload and 10 Mbps download. Thus, the maximum sample rate the proposed approach can adapt is 2400 bytes per second. We rent CAT1 data plans from Telecom Anonymity.

The Cloud: We rent a server in Cloud Anonymity with 8 cores of 2.5 GHz, and a total memory of 128GB. The data in the cloud are stored in XML format.

7 EVALUATION

7.1 Experiment

7.1.1 System Setup. The network topology is shown in Fig. 9. There are three N-nodes and five S-nodes. The links are configured as in the figure. We set S_1 and S_2 to transmit 200 bytes at once every three minutes, and S_3, S_4 and S_5 to transmit 600 bytes at once every minute. We use two pricing models. The first one is provided by China TeleCom, a PAYG-E model, where each 40 MB costs \$1. The second is a MP model where available monthly data plans are shown in Table 1 with \$0.6 charged for each 1 MB exceeding the cap, i.e., $d = 0.6$ in Eq. (1).

We compare three algorithms: 1) Exclusive channel occupation (ECO), the scheme without IoT sharing, 2) FND, and 3) NPS.

7.1.2 Experiment Results. The system is turned on for 3 days and the overall data usage is scaled to one month under the controlled experiment. We derive the overall monthly cost of different schemes. The result is shown in Fig. 10. From Fig. 10(a), we see that in the PAYG-E model, FND leads to a cost saving of 40% as compared with ECO; In Fig. 10(b), for the MP model, NPS leads to a cost saving of 48% as compared with ECO. This matches our expectation since sTube+ TCC link sharing will bring significant cost reductions. Next, we will evaluate different configurations using simulations and we will see that the saving can be more significant when the network is larger.

We now study the operation behavior of sTube+. We run sTube+ for 70 minutes. During this period, we intentionally add some events as shown in Table 2, to emulate node failures, budget exhaustion events, etcetera in N-nodes, etc. We show the operations of sTube+ in Fig. 12. Here we have four sub-charts. The top three charts show the package received and sent by N_1, N_2, and N_3 respectively. The bottom chart shows the residual budget of N_1, N_2 and N_3.

At t_1, N_2 is off. We see that S_3 and S_4, which are originally attached to N_2, switch to $N1$ and N_3 respectively at t_2. At t_3, we turn on N_2. This does not immediately trigger the return of S_3 and S_4, since it is the S-nodes who initiate the peering of S-nodes and N-nodes in our design. At t_4, where N-nodes broadcast their residual budget-indices (rb-indices), all S-nodes independently recompute their peering relationship, and S_3 and S_4 reconnect to N_2. At t_5, the data budget of N_3 is exhausted, and this triggers a recharge of the data budget of N_3 as explained in Section 5. At t_6, the data budget of N_2 is exhausted, and N_2 sends small rb-index to show "low balance", S_3 and S_4 change connections to N_1 and N_3 respectively.

We only observe one packet loss at one event, t_1. The loss occurred because the failure happened before neighbor updates. We conduct 3 days experiment without N-node failure, and we observe that no data loss occurs. We also conduct experiment on the topology only containing S_3, N_1 and N_2. We set the check period to be 1 minutes. We manually turn off N_1 and record the number of the packet loss of S_3 under different sampling period. The result is shown in Fig. 11. We can observe that the number of packet loss is at most one when the sampling period is greater than ten seconds. Note that in practice, the sampling period can be bigger than five seconds. It is also possible to fine-tune the neighbor update interval and check period to reduce packet losses.

Type	Price	Volume
1	3 $	10 MB
2	5 $	50 MB
3	12 $	200 MB
4	32 $	700 MB
5	40 $	1 GB
6	85 $	3 GB
7	135 $	8 GB
8	210 $	15 GB

Table 1: The monthly data plans.

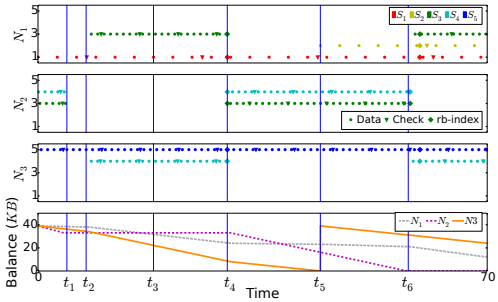

Figure 12: The behavior of FND under the controlled environment.

Time	Event and explanation
t_1	Turn off N_2 to emulate N_2 failure.
t_2	N_2 does not reply to heartbeat messages so that the failure is detected. S_3 and S_4 connect to N_1 and N_3.
t_3	Turn on N_2 to emulate N_2 recovery.
t_4	New rb-index received. Since N_2 has a larger balance, S_3 and S_4 change connections to N_2.
t_5	Even N_3 sends small rb-index to show "low balance", S_5 has to stick to it so that N_3 recharges.
t_6	N_2 sends small rb-index to show "low balance", S_3, S_4 change connections to N_1 and N_3

Table 2: Explanation of the behavior of FND shown in Fig. 12

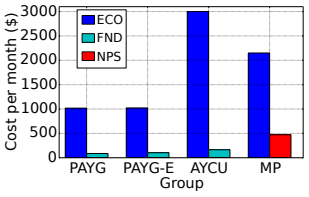

Figure 13: The cost of ECO and sTube+ under different pricing models.

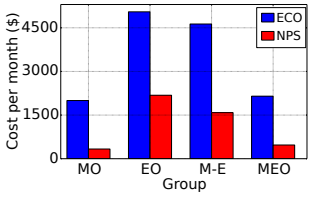

Figure 14: The cost of ECO and NPS under different traffic models.

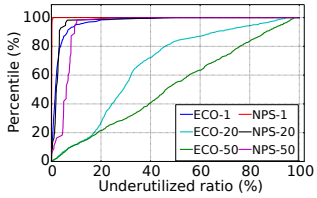

Figure 15: The CDF of underutilized ratio of data volume.

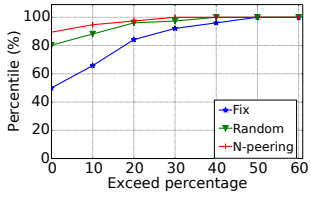

Figure 16: The CDF of the exceed load percentage of N-nodes.

7.2 Simulation

We now use simulations to evaluate sTube+ in large-scale networks and under various parameter settings.

7.2.1 Simulation Setup. We set the network topology by deploying S-nodes and N-nodes randomly and uniformly in a 100×100 m^2 plane. There are 1000 S-nodes and 100 N-node locations. In our tech-report [10], we also evaluate more complicated scenarios where S-nodes and N-nodes are normally distributed leading to similar results. The default data traffic pattern is called Mice and Elephants Only (MEO) where the data volume of 95% S-nodes is uniformly distributed from 1 MB to 3 MB, and the data volume of the rest 5% of S-nodes is uniformly distributed from 30 MB to 50 MB. We will evaluate other traffic patterns in Section 7.2.2 as well.

We study four pricing models: 1) PAYG, the first 40 MB costs \$1, i.e., $L = 40, P_1 = 1$, the prices of the following 40 MB steps are \$0.8, i.e., $P_2, P_3, \ldots = 0.8$; 2) PAYG-E, each 40 MB costs \$1; 3) AYCU, \$3 is charged without data usage limitation; and 4) MP, the monthly data plans are shown in Table 1 with \$0.6 charged for each 1 MB exceeding the cap, i.e., $d = 0.6$.

The default broadcasting frequency of rb-index is set to 10 times per month. We set the default $p_n = 10\%$, $p_{req} = 8\%$.

7.2.2 Simulation Results. We first compare ECO and FND under three PAYG models in Fig. 13. We see that FND shows a much higher cost saving as compared to our experiment results. This matches our expectation since the advantage of sharing becomes more significant when there are more S-nodes to share. FND outperforms PAYG, PAYG-E and AYCU by 91%, 89% and 95% respectively. The AYCU has higher saving compared to PAYG and PAYG-E. This is because the data usage of AYCU is unlimited and more S-nodes can share one TCC link without leading to cost increase. The PAYG has higher saving ratio compared to PAYG-E. The reason is that the price of the step in PAYG becomes cheaper as the TCC link purchases more steps. Thus more S-nodes sharing one TCC link leads to a cheaper average cost and higher saving percentage.

In Fig. 13, we also compare ECO and NPS under the MP pricing model. We see a cost saving of 78% which is less than that of PAYG. This is because, in MP pricing model, the cost gap of two adjacent plans is bigger, thus if the data volume of one monthly data plan can not meet the requirement of a N-node, the N-node should purchase the other one whose price is much higher. while in the PAYG model, the TCC link can purchase steps which are cheaper one by one.

The Impact of Determined Traffic Pattern: We consider four data traffic patterns of S-nodes: 1) Mice Only (MO): the data usage of each S-node is uniformly distributed from 1 MB to 3 MB, 2) Elephants Only (EO): the data usage of each S-node is uniformly distributed from 30 MB to 50 MB, 3) From Mice to Elephants (M-E): the data usage of each S-node is uniformly distributed from 1 MB to 50 MB, and 4) our default MEO.

In Fig. 14, we show the costs of NPS under the aforementioned four traffic patterns in MP pricing model (we also evaluate the overall costs of PAYG, PAYG-E and AYCU under the four traffic patterns in [10]). NPS outperforms ECO by 83%, 57%, 66% and 78% under MO, EO, M-E and MEO respectively. We observe that a greater average data volume of S-nodes will lead to a smaller cost saving gap between NPS and ECO. The reasons are: 1) More nodes can share one step without increasing extra cost if the data volumes of nodes are small; 2) Compared to serving S-nodes with big data volume, sTube+ can serve S-nodes with small data volume well since the fine data volume is easier to be arranged. This illustrates that NPS works effectively under the four traffic patterns.

The Impact of Price Granularity: The granularity means the data volume gap between two adjacent price plans. For example, in our default pricing model, the granularity of prices is 40MB for the PAYG model. It is possible that ISPs can develop more wireless channels with fine-grained prices models. However, to beat the cost of sTube+, ISPs have to develop price models with unrealistic granularity. In Fig. 15, we present the cumulative distribution function

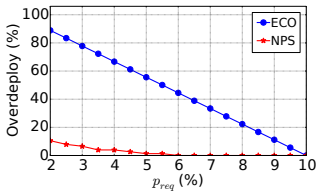

Figure 17: The over-deploy ratio of ECO and NPS as a function of p_{req}.

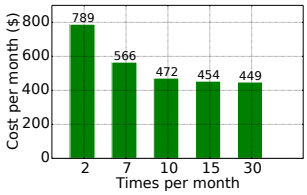

Figure 18: The monthly cost of NPS as a function of update period.

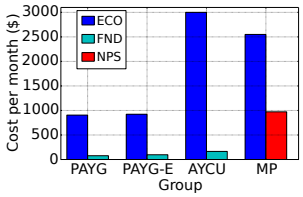

Figure 19: The cost of ECO and sTube+ under dynamic traffic pattern.

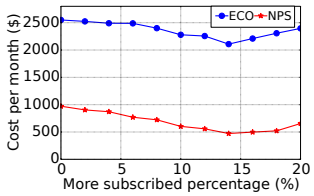

Figure 20: The cost of ECO and sTube+ as function of more subscribed percentage.

(CDF) of the ratios of underutilized data volume of N-nodes under ECO and NPS, when granularities are 1 MB, 20 MB and 50 MB respectively. We can observe that the underutilized data volume ratio of all N-nodes of NPS is under 10% under the 1 MB, 20 MB and 50 MB granularity. For the ECO, the underutilized data volume ratios of N-nodes range from 0% to 100%, only when the granularity is down to 1 MB, most of the N-nodes' underutilized data volume ratios are under 10%. This illustrates that if ECO wants to reach the performance of sTube+, ISPs should provide unrealistic granularity pricing models. On the other hand, the proposed TCC sharing can address this problem without requiring fine granularity.

The Performance of the N-Peering Algorithm: We compare our N-peering algorithm with two algorithms employing NPS for the TCC links. Fixed N-node Connection (Fix): each S-node randomly connects to one active N-node and sticks to it. Periodic Random N-node Connection (Random): each N-node randomly selects one active N-node every update period.

We show the CDF of the exceed percentage of traffic loads of N-nodes in Fig. 16. Please note that the higher exceed percentage will lead to the worse performance, since a higher rate price is charged for each exceeding data usage unit. 80% N-nodes of Random and 90% N-nodes of N-peering do not exceed the subscribed data usage, but the percentile becomes only 49% for Fix. Compared with Fix, we notice that most N-nodes do not use up the subscribed data usage, through using Random and N-peering, with the help of periodic budget updating. Moreover, compared with Random and Fix, the exceed percentage of N-peering is much smaller. Such observations suggest that N-peering is beneficial to balance the traffic loads among N-nodes and distribute the traffic loads with the help of rb-index in more cost-efficient fashions.

The Performance of Over-Deployment on Reliability: In order to meet the required average outage probability p_{req}, sTube+ employs the TCC-OD algorithm (Section 5.4) to over-deploy more N-nodes. We call the ratio between the number of over-deployed N-nodes and the number of N-nodes computed by NPS as *over-deploy ratio*. Fig. 17 shows the required over-deploy ratio as a function of p_{req} under ECO and NPS. We observe that ECO needs to over-deploy much more N-nodes than that of NPS to meet the same p_{req}. When p_{req} is bigger than 6%, NPS is not required to deploy additional N-nodes; when p_{req} is 2%, we only need to over-deploy 10% N-nodes for NPS but 89% for ECO. This illustrates that, to meet the same reliability, much fewer N-nodes should be deployed for sTube+ employing TCC-OD algorithm, compared with ECO.

The Performance on the Update Period: We study the effects of broadcasting frequency of rb-index of N-peering. We observe that the overall cost is decreased by 41% for NPS when the frequency is increased from 2 to 10 each month, as shown in Fig. 18. This illustrates that a reasonably frequent broadcast of rb-index is helpful to further reduce the overall cost. However, the cost reduces less than 5% when the frequency is increased from 10 to 30 each month. This illustrates that too frequent rb-index broadcasting is not necessary as it will not reduce the cost.

The Impact of Dynamic Traffic Pattern: In dynamic traffic pattern, each S-node has a basic monthly data volume with a certain fluctuation. We study the MEO basic traffic pattern and each S-node has an up to 20% fluctuation.

In Fig. 19, we first compare ECO and FND under three PAYG models. We see that FND outperforms PAYG, PAYG-E and AYCU by 90%, 90% and 95% respectively which is similar to the cost saving ration compared to the determined traffic pattern. This is because, under the PAYG pricing model, the data volume is purchased step by step, thus the dynamic traffic pattern has no influence to the cost compared to the determined traffic pattern.

In Fig. 19, we also compare ECO and NPS which subscribes the data volume according to the basic monthly data volume. We see a cost saving of 62% which is less than that of determined traffic pattern. This is because the data volume is subscripted at begin of the billing cycle, and if the purchased data volume cannot cover the usage, the exceeded data volume is charged much higher price. Under the MP pricing model, we can purchase more data volume than the basic monthly data volume. In Fig. 20, we show the cost of ECO and NPS as function of percentage of more data volume purchased than the basic monthly data volume. We see that, if we purchase 14% more of the basic data volume, we can get the least cost. Thus, the cost can be reduced by purchasing more data volume than the basic data volume.

8 A CASE STUDY

We are developing a SAMS for a centralized air-conditioning system (Fig. 21 is an illustration of a centralized air-conditioning system with water tower, chillers to cool down the water, pumps to push water circulation, air handling unit (AHU) to use cold water to cool down the air, and fans to push air circulation. Finally, cold air will air-condition the offices and the temperature is controlled by the amount/speed of cold air allowed into an office).

We compute the performance of a chiller by Coefficient of Performance (COP, COP = $\frac{4.181\times Fr\times(T_r-T_s)}{W_c}$) and the performance of a pump by Water Transfer Coefficient (WTC, WTC = $\frac{Q}{W_p}$) [23, 24].

We develop the sensing module on Raspberry Pi to collect the raw data in Table 3. Chillers and pumps have standard APIs to output data from their embedded sensors. Using chiller as an example, a chiller controller uses a ModBus RTU protocol with an RS-485 interface. Modbus RTU protocol is a query-response protocol. We implement an application in Raspberry Pi using the standard library libmodbus [25] to query the chiller through Modbus RTU protocol.

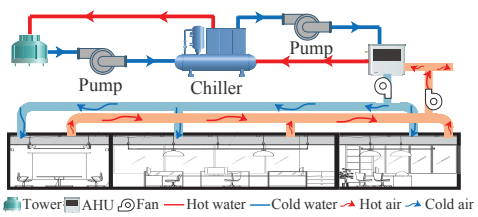

Figure 21: A typical centralized HVAC system.

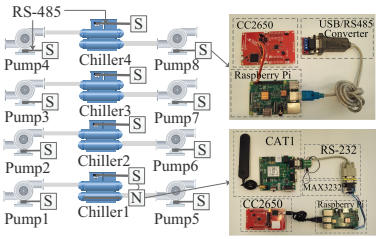

Figure 22: SAMS supported by the sTube+ architecture.

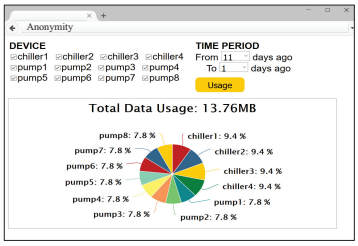

Figure 23: The data usage of the 4 chillers and 8 pumps.

Para.	Description
F_r	Condenser flow rate (m^3/h)
T_r	The returning chilled water temperature ($^\circ C$)
T_s	The supplying chilled water temperature ($^\circ C$)
W_c	Chiller power input (kWh)
Q	Heat transfer to circulating water (kJ)
W_p	Pump power input (kWh)

Table 3: The parameters for computing COP and WTC.

The communication between USB port of Raspberry Pi and RS-485 need a USB/RS485 Converter module as the electrical level difference. Our hardware is shown in Fig. 22.

We deployed one N-node on a chiller and 12 S-nodes, four chillers and eight pumps (Fig. 22). All our nodes are powered by AC and we ran our system for 12 consecutive days. Our cloud monitored the data consumed by each S-node (Fig. 23). Our system can lead to a great cost reduction. In our case, We employ the PAYG-E pricing model provided by China TeleCom. The total data traffic of four chillers and eight pumps in 10 days was 13.76 MB. The monthly communication cost of our system is \$2 . If adopting the ECO method, the cost is \$12 which is six times to our method. Note that, \$2 is also the optimal cost we can get under the real pricing model.

9 CONCLUSION

One core value of the Internet-of-Things is the data of the things (i.e., IoT devices). Yet, transmitting the data to the cloud is still not pervasively achievable. The industry is actively developing various communication choices to support the diverse requirements of IoT data transmission. We demonstrated in this paper, that the number of IoT communication choices may not easily catch up the requirements. We carefully analyzed example application scenarios. We proposed a solution of sTube+ on IoT communication sharing. The design of sTube+ includes a layered data delivery architecture, algorithms for cost optimization, and a prototype of a fully functioning system. We further develop a case study of chiller and pump maintenance, where sTube+ acts as the underlying architecture.

10 ACKNOWLEDGMENTS

Our sincere thanks to the following grants for supporting this work – the University of Sydney DVC Research/Bridging Support Grant; the National Natural Science Foundation of China under Grant 61272464; RGC/GRF under Grant PolyU 5264/13E; the National Science Foundation under grant CNS-1423408.

REFERENCES

[1] Dusit Niyato, Xiao Lu, Ping Wang, Dong In Kim, and Zhu Han. Economics of Internet of Things: an information market approach. *IEEE Wireless Communications*, 23(4):136–145, 2016.

[2] Anna Maria Vegni, Valeria Loscr, Alessandro Neri, and Marco Leo. A Bayesian packet sharing approach for noisy IoT scenarios. In *Proc. IEEE IoTDI'16*, Berlin, Germany, Apr. 2016.

[3] Zimu Zheng, Dan Wang, Jian Pei, Yi Yuan, Cheng Fan, and Fu Xiao. Urban traffic prediction through the second use of inexpensive big data from buildings. In *Proc. ACM CIKM'16*, Indianapolis, IN, Oct. 2016.

[4] Jay Lee, Chao Jin, and Zongchang Liu. Predictive big data analytics and cyber physical systems for tes systems. In *Advances in Through-life Engineering Services*, pages 97–112. Springer, 2017.

[5] Cesar A Garc, Pedro Merino, et al. 3GPP standards to deliver LTE connectivity for IoT. In *Proc. IEEE IoTDI'16*, Berlin, Germany, Apr. 2016.

[6] Ghasem Naddafzadeh-Shirazi, Lutz Lampe, Gustav Vos, and Steve Bennett. Coverage enhancement techniques for machine-to-machine communications over LTE. *IEEE Communications Magazine*, 53(7):192–200, 2015.

[7] JS Roessler. LTE-advanced (3GPP rel. 12) technology introduction. Apr. 2015.

[8] Joseph HK Lai, Francis WH Yik, and Aggie KP Chan. Maintenance cost of chiller plants in hong kong. *Building Services Engineering Research and Technology*, 30(1):65–78, 2009.

[9] Nofirman Firdaus, Bambang Teguh Prasetyo, and Thomas Luciana. Chiller: Performance deterioration and maintenance. *Energy Engineering*, 113(4):55–80, 2016.

[10] Anonymous Authors. stube+: An IoT communication sharing architecture for smart after-sales maintenance in buildings. Technical report, 2017. https://goo.gl/hXEimZ.

[11] Jing Jiang and Yi Qian. Distributed communication architecture for smart grid applications. *IEEE Communications Magazine*, 54(12):60–67, 2016.

[12] Steven Latre, Philip Leroux, Tanguy Coenen, Bart Braem, Pieter Ballon, and Piet Demeester. City of things: An integrated and multi-technology testbed for iot smart city experiments. In *Proc. IEEE ISC2'16*, Trento, Italy, Sep. 2016.

[13] Jingkun Gao, Joern Ploennigs, and Mario Berges. A data-driven meta-data inference framework for building automation systems. In *Proc. ACM Buildsys'15*, Seoul, South Korea, Nov. 2015.

[14] Stephen Dawson-Haggerty, Xiaofan Jiang, Gilman Tolle, Jorge Ortiz, and David Culler. sMAP: a simple measurement and actuation profile for physical information. In *Proc. ACM SenSys'10*, Zurich, Switzerland, Nov. 2010.

[15] Thomas Zachariah, Noah Klugman, Bradford Campbell, Joshua Adkins, Neal Jackson, and Prabal Dutta. The Internet of Things has a gateway problem. In *Proc. ACM HotMobile'15*, Santa Fe, New Mexico, Feb. 2015.

[16] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002.

[17] Mung Chiang and Tao Zhang. Fog and IoT: An overview of research opportunities. *IEEE Internet of Things Journal*, 2016.

[18] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[19] Antonio L Maia Neto, Artur LF Souza, Italo Cunha, et al. Aot: Authentication and access control for the entire iot device life-cycle. In *Proc. ACM Senys'16*, CA, USA, Nov. 2016.

[20] Jianwei Huang and Lin Gao. Wireless network pricing. *Synthesis Lectures on Communication Networks*, 6(2):1–176, 2013.

[21] Petr Slavík. A tight analysis of the greedy algorithm for set cover. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 435–441. ACM, 1996.

[22] Neal E Young. Greedy set-cover algorithms. In *Encyclopedia of algorithms*, pages 379–381. Springer, 2008.

[23] Mehdi Mahdavikhah and Hamid Niazmand. Effects of plate finned heat exchanger parameters on the adsorption chiller performance. *Applied Thermal Engineering*, 50(1):939–949, 2013.

[24] Yang Yao, Yiqiang Jiang, Shiming Deng, and Zuiliang Ma. A study on the performance of the airside heat exchanger under frosting in an air source heat pump water heater/chiller unit. *International Journal of Heat and Mass Transfer*, 47(17):3745–3756, 2004.

[25] A Libmodbus. Modbus library for linux, mac os x, freebsd, qnx and win32. libmodbus-a fast and portable modbus library.