# 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

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.

CCS Concepts: • **Networks** → *Layering*;

Additional Key Words and Phrases: IoT, communication architecture, smart building, thing-to-cloud

**ACM Reference format:**
Chuang Hu, Wei Bao, Dan Wang, Yi Qian, Muqiao Zheng, and Shi Wang. 2018. sTube+: An IoT Communication Sharing Architecture for Smart After-sales Maintenance in Buildings. *ACM Trans. Sen. Netw.* 14, 3–4, Article 29 (November 2018), 29 pages.
https://doi.org/10.1145/3274283



Authors' addresses: C. Hu, QT405, the Hong Kong Polytechnic University, Kwoloo, Hong Kong; email: cschu@comp.polyu.edu.hk; W. Bao, Office 425 (West Wing), School of Information Technologies, Building J12, the University of Sydney, Sydney, Australia; email: wei.bao@sydney.edu.au; D. Wang, PQ708, the Hong Kong Polytechnic University, Kwoloo, Hong Kong; email: csdwang@comp.polyu.edu.hk; Y. Qian, PKI 206B Scott Campus (Omaha), University of Nebraska Lincoln; email: yqian2@unl.edu; M. Zheng, A107, Qianhai Dream Workshop, No.1 qianwan road, Shenzheng, China; email: joe.zheng@fusquare.com; S. Wang, QT415, the Hong Kong Polytechnic University, Kwoloo, Hong Kong; email: winona.wang@connect.polyu.hk.

1550-4859/2018/11-ART29 $15.00
https://doi.org/10.1145/3274283

## 1 INTRODUCTION

One important value proposition of the Internet of Things (IoT) is the data generated by the IoT devices (a.k.a., things) [26]. 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 [30]. 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 [36]. Smart After-sales Maintenance and Services (SAMS), which will become the case study of this article, is another example. Manufacturers of air conditioners, pumps, elevators, and the like, 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 [22].

To fully realize the aforementioned applications, the things should be accessible anywhere and anytime. One key question remains to be answered: how can we 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 [15] 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 [24]. CAT1 was released in 2016 and CAT0 is under deployment [28]. 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 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 Figure 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

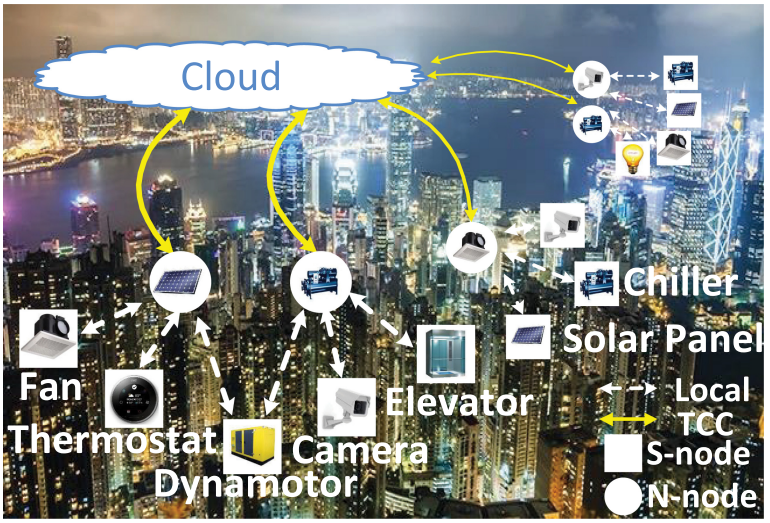


Fig. 1. Smart After-Sales Maintenance Services (SAMS).

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 article 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 (Sections 5 and 6). We prototype a fully functioning system for sTube+ (Section 7).
- We comprehensively evaluate sTube+ (Section 8). In particular, we develop a SAMS case study, using sTube+ as the underlying architecture (Section 9).

## 2 THE MOTIVATION AND RELATED ARCHITECTURE

To ensure that a network architecture is 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 $5,639.94 (we use USD as the monetary unit in the rest of this article) per time [20]. 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

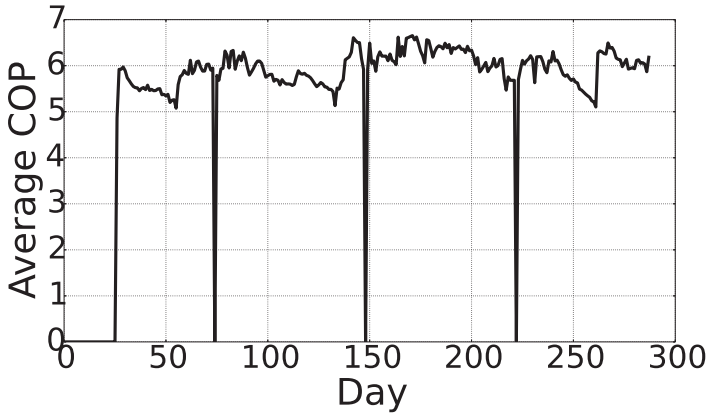


Fig. 2. Average COP of chillers as a function of day.

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) [13]. Maintenance is needed if the COP of a chiller is below a certain threshold.[1] The COP of one chiller measured in our dataset is shown in Figure 2.

To compare the current maintenance plan and SAMS, we obtained 4-year data of 10 chillers in 3 buildings. Let $c_M$ and $c_R$ denote the average cost of each maintenance and emergency repair, then $c_M = 897.12$ and $c_R = 5,639.94$. Let $t, t'$ denote the average number of times of maintenance and emergency repair of the chiller in the 1-year example, respectively. For the routine maintenance, the total cost can be present as $c_{RM} = tc_M + t'c_R$. The SAMS requires an additional communication fee, however, the monitor data rate is low, so for each chiller, the cheapest data usage plan (0.44$ for 10MB) can meet the requirement. Let $c_C$ denote the average monthly communication cost of each chiller, then $c_C = 0.44$. For the SAMS, the total cost can be calculated as $c_{SAMS} = tc_M + 12c_C$.

We calculated the optimal plan for current maintenance with a routine maintenance. Let $T_{RM}$ denote the interval of routine maintenance. We gradually increase $T_{RM}$, compute the corresponding yearly average times of maintenance ($t$) and emergency repair ($t'$) of the sample chillers, and, thus, we can compute the corresponding cost $C_{RM}$ for each $T_{RM}$. We record the optimal (costless) $T_{RM}$. The optimal routine maintenance interval is 3.1 months with 3.87 times maintenance and 0.103 times emergency repair each year. The cost of one chiller of a year for current routine maintenance is $C_{RM} = 3.87 \times 897.12 + 0.103 \times 5,639.94 = \$40,527.64$.

For SAMS, we can collect the chiller data in real time. According to our dataset, the average time it costs for the COP of a chiller reduce to 5.7 after maintenance is 3.89 months; thus, we get the average number of times of maintenance for SAMS is $t = 3.13$ for each year. The SAMS requires an additional communication fee, however, the monitor data rate is low, so for each chiller, the cheapest data usage plan ($0.44 for 10MB) can meet the requirement. The cost of one chiller in a year is $C_{SAMS} = 3.13 \times 897.12 + 12 \times 0.44 = \$28,132.66$. This leads to a 30.58% savings. Note that this is only a baseline comparison. If we consider joint maintenance of multiple equipment,

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

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 [18], 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 45MB. Assume that a piece of equipment has a data volume of 50MB but it can only afford $1. ISPs will not deliberately develop such a plan since it makes CAT1 non-marketable. In a sharing environment, a close-by piece of equipment with residual data of 5MB 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 [21] 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 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 [14, 19]. Traditional BAS are mostly signal-based. An sMap architecture [12] was developed to software-define traditional BAS. In sMap, the IoT devices are organized into a mesh network, and a gateway is used. 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.

**Ad Hoc Network:** A wireless ad hoc network [27] enables devices to create and join networks "on the fly." Each node participates in routing by forwarding data for other nodes, so the determination of which nodes forward data is made dynamically on the basis of network connectivity and the routing algorithm in use. In SAMS, if a node uses ad hoc, we should assume that other S nodes are equipped with ad hoc forwarding functions, which is not the case in many situations. This is because not all devices in the building are powerful enough to run routing functions, as some devices have constraints in computing power, battery life, and memory. In addition, data may be transmitted through multiple hops to the cloud, which increases the risk of packet loss.

**Mobile Phones as Relays:** One recent proposal to transmit IoT data to the cloud is to use mobile phones as relays [35]. 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, an overall optimization of the cost of a vendor, and so on. The level of *complexity* differs greatly. Another example is represented by family plans, where multiple SIM cards are allowed. However, different from the SIM card sharing in mobile data plans, we consider the IoT sharing, so that our scenario is substantially different: (1) Multiple SIM cards sharing data plans require each device to equip with a SIM card. However, owning and managing a SIM card is expensive compared with the limited profit (due to the small amount of data transmitted) at a small IoT device. For example, AT&T charges USD $14.99 for each LTE-Category 1 module (CAT1 SIM card) [3], and Deutsche Telekom Charges €199 for each 25 SIM cards [5]. A SIM card is affordable for the expensive mobile phone, while it may be overcostly for the majority of the IoT devices. In proposed sTube+, only parts of devices (i.e., N-node) should equip with SIM cards, which can reduce the cost significantly compared with each device owning a dedicated SIM card. (2) The monthly data volume used by a phone is much greater than that of a single IoT device. The ISP is unwilling to provide a small data volume plan for the profit reason [14], for example, the data plan provided by China Telecom for NB-IoT charges according to the connection frequency [4]. The objective of sTube+ is to organize a greater number of IoT devices to share fewer choices of TCC links in a cost-efficient way. (3) For the profit of ISP itself, the sharing scale ISP supposes is small. For example, only up to 3 and 10 SIM cards is admitted by Company Single [1] and $O_2$ [2], respectively, to share one data plan, while the devices of a vendor in the building may be in the scale of a thousand. Since the large-scale sharing is not allowed by ISPs, the users themselves are motivated to establish a new framework to share the large number of devices. The proposed sTube+ supposes a larger communication sharing scale in the user side. As such, we believe that ISPs will impose certain *limits,* 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) [6] and **Fog Computing** [10, 11]. 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 LOC links that are local and free. There are three types of nodes (see Figure 1): sensing nodes (connected to the SAMS equipment), nodes with TCC links, and the cloud servers. In this article, we call them S-nodes, N-nodes, and the clouds. Note that S-nodes and N-nodes 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, we must 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, and so on.

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.

The cloud runs the centralized price optimization algorithms, outputs the adopted data plan and the corresponding subscripted data volume of each N-node, and transforms the result to the N-node, respectively. According to the subscripted data volume of N-nodes, S-nodes and N-nodes run the distributed network control algorithm to control the data flow from the S-node to the cloud (details in Section 4.1.2).

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 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 a piece of 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 to be 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 Figure 3.

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.

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

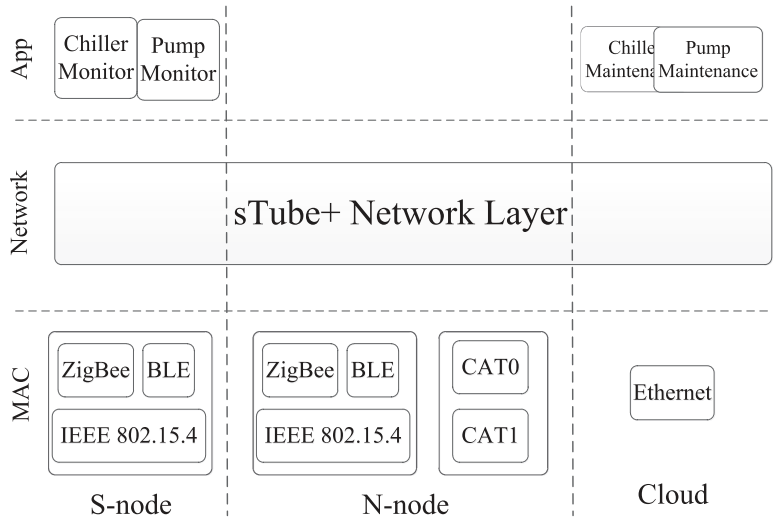


Fig. 3. A layered architecture.

For a very small-scale network, the cloud can adopt a centralized design, where it computes all connections and broadcasts to the S-nodes and N-nodes 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, and so on. 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 article, 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 with the N-node peering process in Figure 4.

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

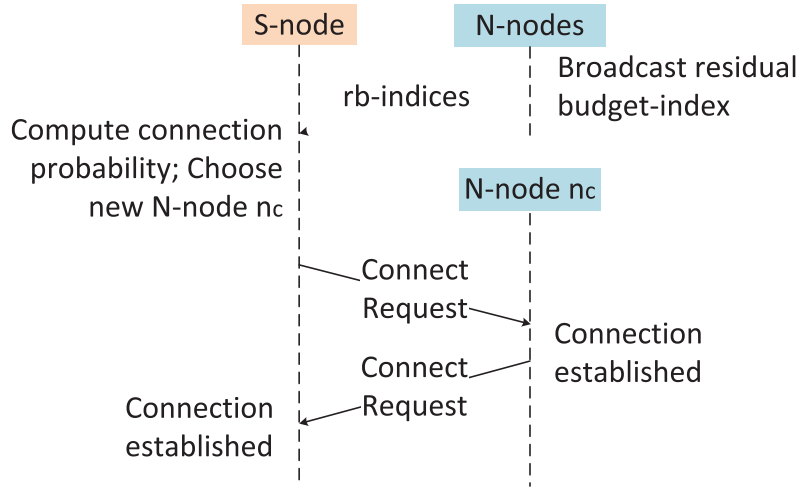


Fig. 4. Illustration of nodes interaction when periodically choosing N-node.

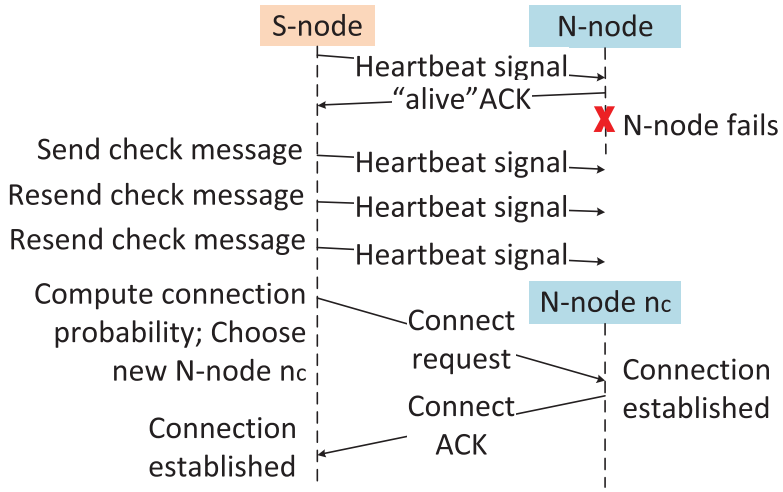


Fig. 5. Illustration of nodes interaction when the connected N-node is failure.

**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 its neighbor if the original neighbor fails. The process is shown in Figure 5. Specifically, the S-node periodically sends a heartbeat signal to the connected N-node and waits "alive" acknowledgement (ACK) from it. If the S-node does not receive a response from the N-node in time $t$, it resends a heartbeat signal to the connected N-node. If the S-node does not receive any response after three heart signals, the S-node regards the N-node as a failure and updates to a new neighbor. To minimize possible data loss, the S-nodes with a higher data rate 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 the Constrained Application Protocol (CoAP) (described in Section 7.1), the data loss of an S-node occurs only when the connected N-node fails.

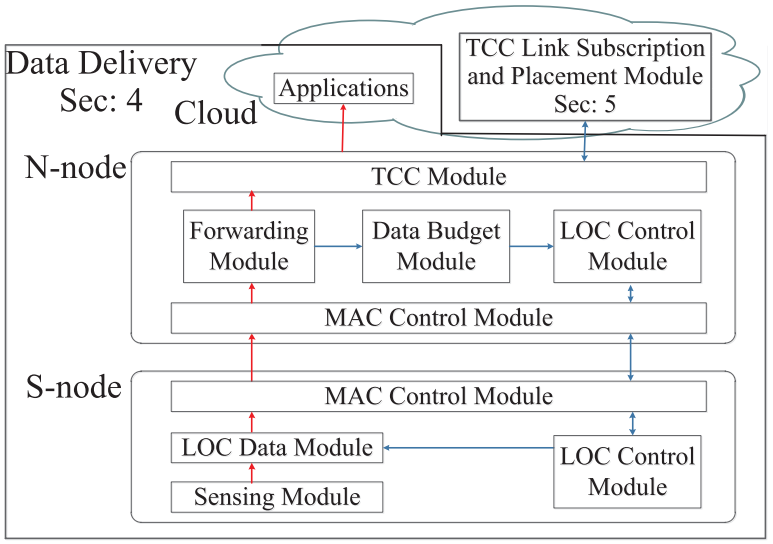


Fig. 6. sTube+ module design.

As the very first work, we assume in this article that (1) our context is one vendor only, (2) N-nodes will not relay traffic for other N-nodes, (3) S-nodes will not relay traffic for other S-nodes, and (4) there is no in-network processing of the traffic. We plan to study routing of other forms, intermediate traffic caching, coding, and so on, and multi-vendor joint optimization in our future work.

### 4.2 Detailed Modules for a Functioning System

A functioning system has modules of all layers; see Figure 6.

**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 Refs [9, 25]. 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 how the S-node and N-node can verify each other's public key when the 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 manufacturer's private key) when the node is produced, i.e., certificate pinning. The manufacturer'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 $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$ denote the set of possible locations of N-nodes. Note that not all devices can act as N-nodes by considering the power of the devices. Only the devices that are equipped with external power source and certain hardwares (e.g., memory and computing power) can be candidates of the N-node. 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 $\mathcal{S} = \{s_1, s_2, \ldots, s_M\}$ 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, the price is represented by Equation (1). This is a staircase function. Here, $x$ is the data usage (i.e., $U_j$); $L$ is an integer to denote the step size of pricing model. Let $P_i$ be the price for the $i$-th step of $L$ data volume. In practice, $P_i$ decreases as the price step increases [17] and $\lim_{i \to +\infty} P_i = P_{min}$, where $P_{min}$ is positive.

$$C_{PAYG}(x) = \sum_{i=1}^{\lceil \frac{x}{L} \rceil} P_i. \tag{1}$$

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 price is represented by Equation (2). Here, $a$ denotes the price of each step. The PAYG-E is what is provided by Telecom in our experiments.

$$C_{PAYG-E}(x) = a \left\lceil \frac{x}{L} \right\rceil. \tag{2}$$

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 Equation (3). 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. For example, the data plan may specify that the first 1GB of data per month is charged at a flat price of $20. Any additional data above that will cost $1 for each 1MB.

$$C_{m_h}(x) = \begin{cases} c_h & , x \le k_h, \\ c_h + d(x - k_h) & , x > k_h. \end{cases} \tag{3}$$

*5.1.3 Problem Formulation.* The overall monetary cost using PAYG is $\sum_{j=1}^{N} C_{PAYG}(\sum_{i:s_i \in \mathcal{S}_j} u_{ij})$. We assume that each $s_i$ can choose best peering $n_j$ to transfer a certain amount of its data. Thus, we arrive at the following problem:

Problem 1 (PAYG TCC Sharing). *Determine the placement of N-nodes $f(n_j)$ and the amount of data uploaded via N-nodes for each S-node $u_{ij}$, $\forall i, j$, subject to (1) $\sum_{j:f(n_j)=1 \text{ and } s_i \in \mathcal{S}_j} u_{ij} = u_i, \forall i$, (2) $u_{ij} = 0$, if $s_i \notin \mathcal{S}_j$ or $f(n_j) = 0$, to minimize the sum PAYG costs over all N-nodes $\sum_{j=1}^{N} C_{PAYG}(\sum_{i:s_i \in \mathcal{S}_j} u_{ij})$.*

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 $n_j$. We have the following problem:

Problem 2 (MP TCC Sharing). *Determine the placement of N-nodes $f(n_j)$, the amount of data uploaded via N-nodes for each S-node $u_{ij}$, and the employed data plan $d'_j$, $\forall i, j$, subject to (1) $\sum_{j:f(n_j)=1 \text{ and } s_i \in \mathcal{S}_j} u_{ij} = u_i, \forall i$, (2) $u_{ij} = 0$, if $s_i \notin \mathcal{S}_j$ or $f(n_j) = 0$, to minimize the sum MP cost over all N-nodes $\sum_{j=1}^{N} C_{d'_j}(\sum_{i:s_i \in \mathcal{S}_j} u_{ij})$.*

*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 [29]. For the PAYG TCC Sharing Problem, we consider a special case that $u_i = 1, \forall i$, and $L$ is sufficiently large, $P_1 = P_{min} = a$, so that the monetary cost at each N-node is $a$ if it is active and is 0 if inactive. Therefore, we aim to minimize the number of active N-nodes. As a result, the PAYG TCC Sharing Problem is equivalent to an optimal set cover problem: to select a minimum number of sets from $\{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_N\}$ that covers all elements in the universe $\mathcal{S}$.

For the MP TCC Sharing Problem, we consider a special case that $u_i = 1, \forall i$, and only a 1-month plan with unlimited data volume and the cost of $C$ is provided, so that the monetary cost at each N-node is $C$ if it is active and is 0 if inactive. Therefore, we aim to minimize the number of active N-nodes. As a result, the MP TCC Sharing Problem is equivalent to an optimal set cover problem: to select a minimum number of sets from $\{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_N\}$ that covers all elements in the universe $\mathcal{S}$. The set cover problem is NP-complete, so Problems PAYG TCC Sharing and MP TCC Sharing are both NP-complete. □

The complexity of exhaustive search for solving the whole problem is $\Theta((P+1)^N)$, where $N$ is the number of possible N-nodes and $P$ is the number of data plans. 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 Ref. [29]. For price subscription, we develop a simple algorithm where every N-node subscribes $P_1$, i.e., the first step price, and recharge $P_l$ ($l = 2, 3, \ldots$) if necessary. We call this algorithm Fast N-node Deployment (FND).

Lemma 1. *In the PAYG model, assume we have an optimal coverage. Then FND is optimal.*

The optimality is because in the PAYG model, the price of each step decreases as the step increases, i.e., a convex function. Again, we see that the complexity comes from the coverage.

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.* We now show that FND is an approximation algorithm for PAYG TCC Sharing with an approximation ratio of $\frac{2P_1^2}{P_{min}^2}(\ln M + 1)$.

Let the performance of FND be $r_{\text{fnd}}$, and the optimal performance of PAYG TCC sharing be $r_{\text{opt}}$. We aim to bound the approximation ratio $\frac{r_{\text{fnd}}}{r_{\text{opt}}}$. Directly bounding $\frac{r_{\text{fnd}}}{r_{\text{opt}}}$ is challenging. We introduce an auxiliary performance, $r_{\text{osc}}$, which is the monetary cost led by an optimal set cover algorithm of S-nodes. Through the auxiliary algorithm, the number of installed N-nodes is minimized.

We can bound the ratios of $\frac{r_{\text{fnd}}}{r_{\text{osc}}}$ and $\frac{r_{\text{osc}}}{r_{\text{opt}}}$.

Lemma 2. *The ratio of the $\frac{r_{fnd}}{r_{osc}}$ is bounded by $\frac{P_1}{P_{min}}(\ln M + 1)$.*

Proof. Let $K_h, K_o$ denote the number of sets of FND and the optimal set cover, respectively. In Ref. [34], it proves $\frac{K_h}{K_o} \le \ln M + 1$. Let $X$ denote the total data usage of all S-nodes, and $X = LQ + R$, where $Q \in \mathbb{N}, 0 \le R < L$. Let $x_i$ ($i = 1, 2, \ldots, K_h$) denote the data usage of set $i$ of FND, and $x_i = Lq_i + r_i$, where $q_i \in \mathbb{N}, 0 \le r_i < L$. Thus, we have $\sum_{i=1}^{K_h} x_i = X$. Let $l(y)$ denote the indicator function to indicate if $y$ is 0. $l(y) = 0$ if $y = 0$; $l(y) = 1$ if $y > 0$. We consider the following two cases.

**Case 1:** $x \ge LK_o, Q \ge K_o$. It is easy to prove that $\sum_{i=0}^{k_h} r_i \ge R$ by reduction to absurdity. $r_{fnd} = \sum_{i=1}^{K_h}\sum_{j=1}^{q_i} P_j + \sum_{i=1}^{K_h} P_{q_i+1} l(r_i)$. $r_{osc} = \sum_{i=1}^{Q} P_i + P_{Q+1} l(R)$. Thus, we have: $\frac{r_{fnd}}{r_{osc}} = \frac{\sum_{i=1}^{K_h}\sum_{j=1}^{q_i} P_j + \sum_{i=1}^{K_h} P_{q_i+1} l(r_i)}{\sum_{i=1}^{Q} P_i + P_{Q+1} l(R)} \le \frac{P_1}{P_{min}} \frac{\frac{X - \sum_{i=1}^{K_h} r_i}{L} + \sum_{i=1}^{K_h} l(r_i)}{Q + l(R)} \le \frac{P_1}{P_{min}}\left(\frac{\frac{X-R}{L}}{Q+l(R)} + \frac{\sum_{i=1}^{K_h} l(r_i)}{Q+l(R)}\right) \le \frac{P_1}{P_{min}}\left(1 + \frac{K_h}{K_o}\right) \le \frac{P_1}{P_{min}}(\ln M + 1)$.

**Case 2:** $X < LK_o, X > K_h$. $r_{osc} \ge P_1 K_o$. $r_{fnd} \le P_1 \lceil \frac{X-(K_h-1)}{L} \rceil + K_h - 1 \le P_1 \lceil \frac{LK_o-(K_h-1)}{L} \rceil + K_h - 1$. Thus, $\frac{r_{fnd}}{r_{osc}} \le \lceil \frac{LK_o-(K_h-1)}{LK_o} \rceil + \frac{K_h-1}{K_o} \le 1 + \frac{K_h}{K_o} \le \ln M + 1$.

Case 1 and Case 2 are exhaustive. The induction is complete. □

Lemma 3. *The ratio of the $\frac{r_{osc}}{r_{opt}}$ is bounded by $\frac{2P_1}{P_{min}}$.*

Proof. Let $K_o$ denote the number of sets of the optimal set cover. Let $X$ denote the total data usage of all S-nodes, and $X = LQ + R$, where $Q \in \mathbb{N}, 0 \le R < L$. Let $x_i$ ($i = 1, 2, \ldots, K_h$) denote the data usage of set $i$ of FND, and $x_i = Lq_i + r_i$, where $q_i \in \mathbb{N}, 0 \le r_i < L$. Thus, we have $\sum_{i=1}^{K_o} x_i = X$. Thus, we have $\sum_{i=1}^{K_o} x_i = X$. It is easy to prove that $\sum_{i=0}^{k_o} r_i \ge R$ and $\sum_{i=0}^{k_o} q_i \le Q$

by reduction to absurdity. Let $l(y)$ denote the indicator function to indicate if $y$ is 0. $l(y) = 0$ if $y = 0$; $l(y) = 1$ if $y > 0$. Thus, $r_{osc} = \sum_{i=1}^{K_o} C_{PAYG}(x_i) = \sum_{i=1}^{K_o} \sum_{j=1}^{q_i} P_j + \sum_{i=1}^{K_o} P_{q_i+1} l(r_i) \leq P_1(\sum_{i=1}^{K_o} \sum_{j=1}^{q_i} P_1 + \sum_{i=1}^{K_o} P_l(r_i)) \leq (Q + K_o)P_1$. The optimal cost plan fills up the first step of the $K_o$ sets and the reminder data usages are contained by one set. Then, we have $r_{opt} \geq K_o P_1 + C_{PAYG}(X - (K_o - 1)L) \geq K_o P_1 + \sum_{i=2}^{X-(K_o-1)L} P_i = K_o P_1 + \sum_{i=2}^{\lceil \frac{X}{L} \rceil - K_o} P_i \geq \lceil \frac{X}{L} \rceil P_{min}$. Thus, $\frac{r_{osc}}{r_{opt}} \leq \frac{(Q+K_o)P_1}{\lceil \frac{X}{L} \rceil P_{min}} = \frac{P_1}{P_{min}} \frac{Q+K_o}{\lceil \frac{X}{L} \rceil} \leq \frac{P_1}{P_{min}} \frac{Q+K_o}{Q+\lceil \frac{K_o}{L} \rceil} \leq \frac{P_1}{P_{min}} \frac{2Q}{Q+\lceil \frac{K_o}{L} \rceil} \leq \frac{2P_1}{P_{min}}$.

The induction is complete. □

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

Please note that the factor $\ln M$ stems from the greedy set cover algorithm [34]. 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: $\quad [N', \mathcal{Z}] \leftarrow$ Node-Partition($\mathcal{Z}, N'$)
5: $\quad [minset, mincost] \leftarrow$ 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 the 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 these 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, and the like 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 $p_{\text{req}}$ 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 $p_{\text{out}}$ under the current network topology. If $p_{\text{out}} < p_{\text{req}}$, we iteratively install one additional N-node that can maximally decrease $p_{\text{out}}$.

## 6 INCREMENTAL DEPLOYMENT OF S-NODE

We now specifically consider the incremental deployment of S-nodes. In practice, a vendor may deploy their products into a building according to a strategy. The strategy does not allow to deploy a certain number of S-nodes into a building at one time and requires to deploy these S-nodes step by step during a long period, due to the negotiation of building management, incremental purchasing by the building and so on. The vendor may know the possible S-nodes that will be deployed in a building over a long period, but it does not know which S-nodes will be deployed in which billing cycle and the deploying sequence of the S-nodes. We formulate the TCC link sharing problem under the incremental deployment of S-nodes according to the PAYG pricing model. We show this problem is NP-complete and propose approximation algorithms.

### 6.1 Problem

Let $\mathcal{S}' = \{s'_1, s'_2, \ldots, s'_{M'}\}$ be the set of the possible deploying S-nodes. Let $Q$ be the deploying sequence of the S-nodes in a billing cycle. The elements of $\mathcal{S}'$, the locations of N-nodes $\mathcal{N}$, and the subset of S-nodes $\mathcal{S}_j$ that can reach N-node $n_j$ are known in advance; however, the set of deploying S-nodes $Q$ given is not known in advance.

We have the following PAYG TCC Sharing under Incremental Deployment (PAYG-TS-ID) problem:

Problem 4 (PAYG-TS-ID). *Given the locations of N-nodes $\mathcal{N}$, the possible deploying S-nodes $\mathcal{S}'$, the monthly data volume of S-nodes $s_i$, and the deploying sequence of S-node $Q$, determine the placement of N-nodes $f(n_j)$ and the amount of data uploaded via N-nodes for each S-node $u_{ij}$, to minimized the sum PAYG cost over all N-nodes $\sum_{j=1}^{N} C_{PAYG}(\sum_{i:s_i \in \mathcal{S}_j} u_{ij})$.*

Theorem 3. *Problem PAYG-TS-ID is NP-complete.*

Proof. To show the problem PAYG-TS-ID is NP-complete, we reduce the online set cover Problem in Ref. [7] to it. The former is proven NP-complete in Ref. [8]. The proven theorem is viewed as follows: Given a ground set of $M'$ elements $\mathcal{Y} = \{y_1, y_2, \ldots, y_{M'}\}$, $\mathcal{Y}$'s family of subsets $\mathcal{T}$, $|\mathcal{T}| = N$, consider a game between an algorithm and an adversary. An adversary gives elements to the algorithm from $\mathcal{Y}$ one by one. Once a new element is given, the algorithm has to cover it by some set of $\mathcal{Y}$ containing it. Denote by $\mathcal{Y}' \subset \mathcal{Y}$ the set of elements given by the adversary. It is NP-complete to minimize the number of the sets chosen by the algorithm.

We consider the following special case: all S-nodes have equal monthly data volume, and the PAYG pricing model is an all-you-can-use pricing model. Now, we can regard the set of the possible deploying S-node $\mathcal{S}'$ as the ground set $\mathcal{Y}$ in the online set cover, the sequence of deploying S-node $Q$ as the set of elements $\mathcal{Y}'$ given by the adversary, and the subset of S-nodes $\mathcal{S}_j$ that can reach N-nodes $n_j$ as the element of $\mathcal{T}$. As a result, the PAYG-TS-ID Problem is equivalent to an optimal online set cover problem: to select a minimum number of sets from $\{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_N\}$ that covers all elements $\mathcal{Y}'$ given by the adversary from the universe $\mathcal{Y}$. □

### 6.2 The Approximation Algorithm for PAYG-TS-ID

*6.2.1 The Algorithm.* The PAYG-TS-ID problem can be divided into two subproblems, i.e., TCC link placement and TCC link subscription as described in Section 5.2.1. We develop an online N-node placement and subscription (ONPS) algorithm for PAYG-TS-ID. The ONPS algorithm contains two subfunctions `online-node-place()` and `node-subscription()`. The subfunction `online-node-place()` adopts the online algorithm in Ref. [8] to determine the placement of N-nodes according to the sequence of deploying S-nodes $\boldsymbol{Q}$. The subfunction `node-subscription()` subscribes $P_1$, i.e., the first step price for every N-node and recharge $P_l (l = 2, 3, \ldots)$, if necessary.

*6.2.2 Approximation Ratio Analysis.* We now show that ONPS is an approximation algorithm for PAYG-TS-ID with an approximation ratio of $\frac{P_1}{P_{min}}(\frac{\log M' \log N}{\log\log M' + \log\log N} + 1)$.

Theorem 6.1. *The approximation ratio of ONPS for PAYG-TS-ID is* $\frac{P_1}{P_{min}}(\frac{\log M' \log N}{\log\log M' + \log\log N} + 1)$.

Proof. Let the cost of ONPS be $r_\mathrm{f}$, and the optimal cost of PAYG-TS-ID (denoted as OPT) be $r_\mathrm{o}$. Directly proving $\frac{r_\mathrm{f}}{r_\mathrm{o}} \le \frac{P_1}{P_{min}}(\frac{\log M' \log N}{\log\log M' + \log\log N} + 1)$ is hard; we prove this by divide and conquer. As the cost is related to the number of installed N-nodes and the number of purchased data volume steps under the PAYG pricing model, we first prove the installed N-node number of ONPS (denoted as $k_\mathrm{f}$) and OPT (denoted as $k_\mathrm{o}$) meets $\frac{k_\mathrm{f}}{k_\mathrm{o}} \le \frac{\log M' \log N}{\log\log M' + \log\log N}$; then, we prove the purchased steps of ONPS (denoted as $t_\mathrm{f}$) and OPT (denoted as $t_\mathrm{o}$) meets $t_\mathrm{f} \le t_\mathrm{o} + k_\mathrm{f}$. Based on these, we can prove $\frac{r_\mathrm{f}}{r_\mathrm{o}} \le \frac{P_1}{P_{min}}(\frac{\log M' \log N}{\log\log M' + \log\log N} + 1)$.

We first prove $\frac{k_\mathrm{f}}{k_\mathrm{o}} \le \frac{\log M' \log N}{\log\log M' + \log\log N}$. Let $k_\mathrm{min}$ be the minimum number of N-nodes that can cover all S-nodes. We have $\frac{k_\mathrm{min}}{k_\mathrm{o}} \le 1$ and $\frac{k_\mathrm{f}}{k_\mathrm{min}} \le \frac{\log M' \log N}{\log\log M' + \log\log N}$ (by the approximation ratio of the online set cover algorithm in Ref. [8]). Then, we have $\frac{k_\mathrm{f}}{k_\mathrm{o}} \le \frac{\log M' \log N}{\log\log M' + \log\log N}$.

We then prove $t_\mathrm{f} \le t_\mathrm{o} + k_\mathrm{f}$. Let $X$ denote the total data usage of all S-nodes, and $x = Lq - r$, where $q \in \mathbb{N}, 0 \le r < L$. $q \le t_f$, $q \le t_o$ (otherwise, the purchased data volume of FND and OPT is smaller than total data usage $x$). Let $x_j$ $(j = 1, 2, \ldots, k_\mathrm{f})$ denote the data usage of the installed N-node $j$ of ONPS, and $x_j = Le_j - r_j$, where $e_j \in \mathbb{N}, 0 \le r_j < L$. $e_j$ is the steps purchased by the installed N-node $j$; thus, $t_\mathrm{f} = \sum_{j=1}^{k_\mathrm{f}} e_j$, $x = qL - r = L\sum_{j=1}^{k_\mathrm{f}} e_j - \sum_{j=1}^{k_\mathrm{f}} r_j$. We have $t_\mathrm{f} = q - \frac{r}{L} + \frac{\sum_{j=1}^{k_\mathrm{f}} r_j}{L}$, as $0 \le r_j < L$, and we have $t_\mathrm{f} \le q + \frac{k_\mathrm{f} L}{L} = q + k_\mathrm{f}$.

At last, we prove $\frac{r_\mathrm{f}}{r_\mathrm{o}} \le \frac{P_1}{P_\mathrm{min}}(\frac{\log M' \log N}{\log\log M' + \log\log N} + 1)$ based on the above proof. The price of each step ranges from $p_1$ to $p_\mathrm{min}$; thus, $r_\mathrm{f} \le p_1 t_\mathrm{f}$ and $r_\mathrm{o} \ge p_{min} t_\mathrm{o}$. $\frac{r_\mathrm{f}}{r_\mathrm{o}} \le \frac{p_1 t_\mathrm{f}}{p_\mathrm{min} t_\mathrm{o}} \le \frac{p_1}{p_\mathrm{min}}(\frac{t_o}{t_o} + \frac{k_f}{t_o})$, each installed N-node must purchase at least one step, thus $t_o \le k_o$, so $\frac{r_\mathrm{f}}{r_\mathrm{o}} \le \frac{p_1}{p_\mathrm{min}}(1 + \frac{k_\mathrm{f}}{k_\mathrm{o}}) \le \frac{P_1}{P_\mathrm{min}}(\frac{\log M' \log N}{\log\log M' + \log\log N} + 1)$. □

## 7 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 9, where we develop the sensing module to collect data from real equipment and conduct data analytics in the cloud.

### 7.1 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.

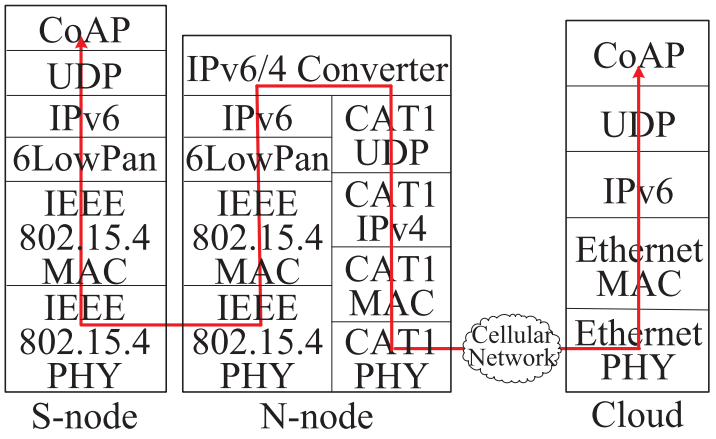


Fig. 7. End-to-End Communication.

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 Figure 7), 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 addresses 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 through 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 an 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 S-node will automatically retransmit a confirmable message if an acknowledgment message is not received in the timeout interval.

### 7.2 The Routing Choice

In our IoT application context, data are routed from the S-nodes to the cloud. We choose OPv6 Routing Protocol for Low-Power and Lossy Networks (RPL) [32] for routing. RPL is a gradient routing technique that organizes nodes as a Direct Acyclic Graph (DAG) rooted at the sink. RPL has an objective function. The goal is to minimize the cost to reach the sink from any node. This function has to be customized. Recall that in our algorithm, we compute the amount of traffic an S-node sends to each peering N-node. In our implementation, the objective function maintains a "volume-N-node" table. The table records the residual data volume of the S-node that can be transmitted through its peering N-node. The objective function chooses the N-node with residual data volume in a round-robin fashion.

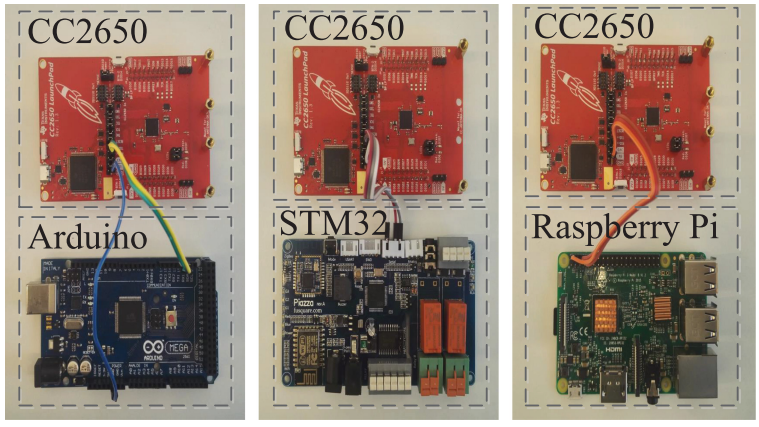


Fig. 8. The S-node.

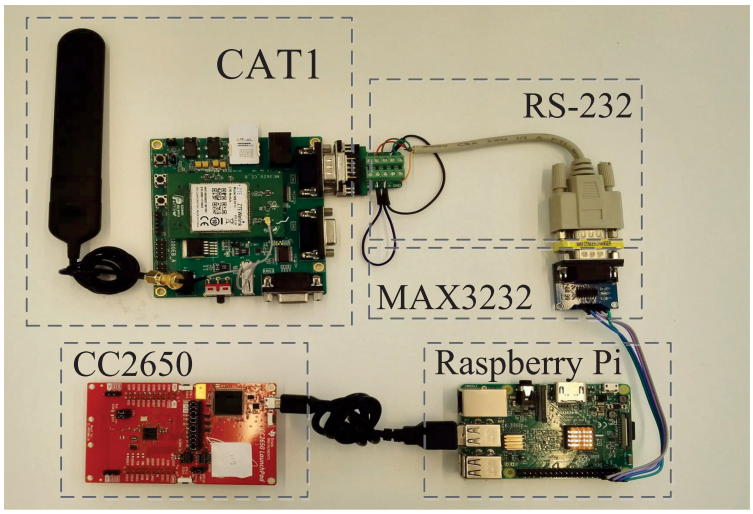


Fig. 9. The N-node.

### 7.3 Hardware Choices

**The S-nodes:** We use Arduino MEGA 2560, STM32, and Raspberry Pi 3 Model B as the S-node hardware board (Figure 8) by considering the different requirements of the capability of the hardware board from the equipment and the hardware cost. For example, for the Fan, only the speed of the 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 the 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 (Figure 9). For LOC side, we use a Texas Instruments CC 2560 SimpleLink$^{TM}$ Wireless MCU for the 802.15.4 radio interface.

Table 1. The Monthly Data Plans

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

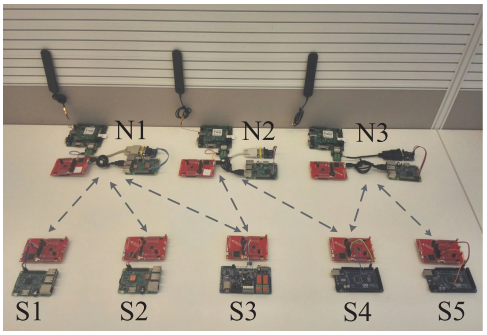


Fig. 10. The network topology of the experiments.

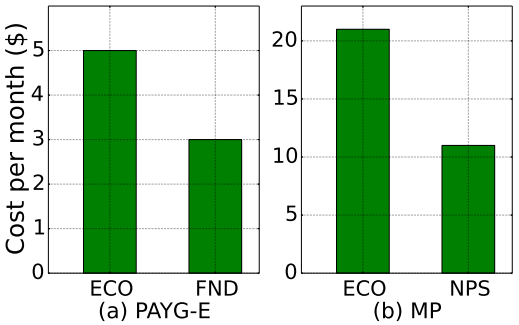


Fig. 11. The monthly cost of different schemes.

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 Transistor-Transistor Logic (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 19,200 bits per second, i.e., 2,400 bytes per second. The CAT1 module supports speeds of 5Mbps upload and 10Mbps download. Thus, the maximum sample rate the proposed approach can adapt is 2,400 bytes per second. We rent CAT1 data plans from Telecom Anonymity.

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

## 8 EVALUATION

### 8.1 Experiment

*8.1.1 System Setup.* The network topology is shown in Figure 10. 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 3 minutes, and $S_3$, $S_4$, and $S_5$ to transmit 600 bytes at once every minute. For the incremental deployment of S-node, we deploy S-node from $S_1$ to $S_5$ one by one. We use two pricing models. The first one is provided by China TeleCom, a PAYG-E model, where each 40MB costs $1. The second is a MP model where available monthly data plans are shown in Table 1 with $0.60 charged for each 1MB exceeding the cap, i.e., $d = 0.6$ in Equation (3).

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

*8.1.2 Experiment Results.* The system is turned on for 7 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 Figure 11. From Figure 11(a), we see that in the PAYG-E model, FND leads to a cost saving of 40% as compared with ECO; in Figure 11(b), for the MP model, NPS leads to a cost savings of 48% as compared with ECO.

For the incremental deployment of S-node, we compare the following three algorithms: (1) ECO, (2) Random connection (RC), where the new S-node chooses an N-node within its range randomly

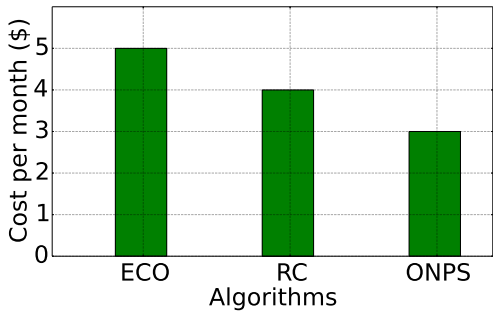


Fig. 12. The monthly cost of incremental deployment of S-node under different schemes.

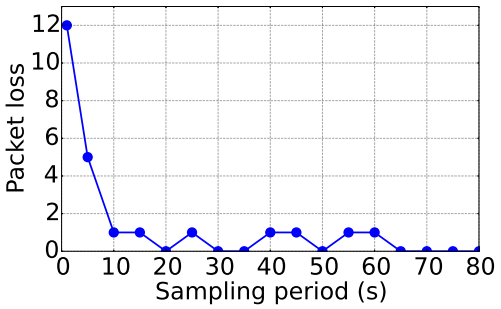


Fig. 13. The packet loss as function of sampling period.

Table 2. Explanation of the Behavior of FND Shown in Figure 14

| 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 $N_3$ |

to connect, (3) ONPS. The overall monthly cost result of the incremental deployment is shown in Figure 12. From Figure 12, we see that under the PAYG-E model, ONPS outperforms ECO and RC by 40% and 25%, respectively. 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 savings 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, in N-nodes, and so on. We show the operations of sTube+ in Figure 14. 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," and $S_3$ 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 of experiments without N-node failure, and we

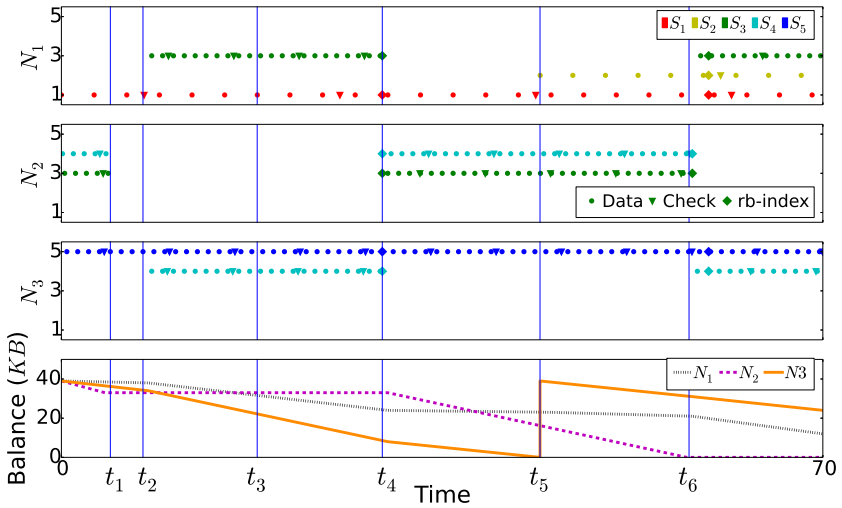


Fig. 14. The behavior of FND under the controlled environment.

observe that no data loss occurs. We also conduct experiments on the topology only containing $S_3$, $N_1$, and $N_2$. We set the check period to be 1 minute. We manually turn off $N_1$ and record the number of the packet loss of $S_3$ under a different sampling period. The result is shown in Figure 13. We can observe that the number of packet losses is at most one when the sampling period is greater than 10 seconds. Note that, in practice, the sampling period can be bigger than 5 seconds. It is also possible to fine-tune the neighbor update interval and check period to reduce packet losses.

## 8.2 Simulation

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

*8.2.1 Simulation Setup.* We set the network topology by deploying S-nodes and N-nodes randomly and uniformly in a $100 \times 100$ m$^2$ plane. There are 1,000 S-nodes and 100 N-node locations. In our tech-report [16], 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% of S-nodes is uniformly distributed from 1MB to 3MB, and the data volume of the rest of the 5% of S-nodes is uniformly distributed from 30MB to 50MB. We will evaluate other traffic patterns in Section 8.2.2 as well.

We study four pricing models: (1) PAYG, the first 40MB costs \$1, i.e., $L = 40, P_1 = 1$, and the prices of the following 40MB steps are \$0.8, i.e., $P_2, P_3, \ldots = 0.8$; (2) PAYG-E, each 40MB 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 1MB 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 $p_n = 10\%, p_{req} = 8\%$.

*8.2.2 Simulation Results.* We first compare ECO and FND under three PAYG models in Figure 15. We see that FND shows a much higher cost savings 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 savings compared to PAYG and PAYG-E. This is

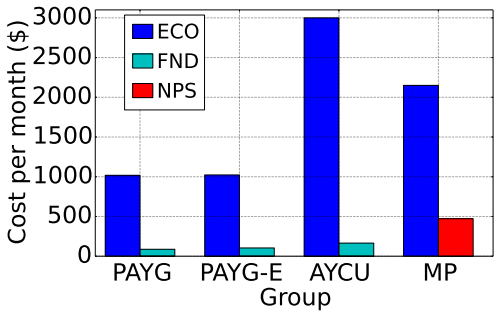


Fig. 15. The cost of ECO and sTube+ under different pricing models.

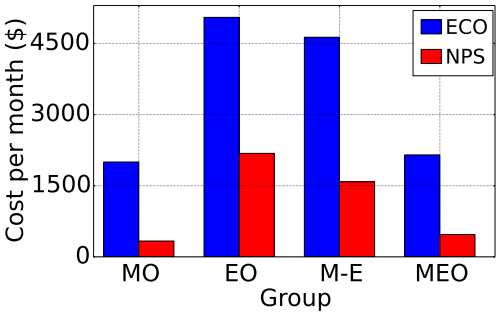


Fig. 16. The cost of ECO and NPS under different traffic models.

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 a higher savings 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 savings percentage.

In Figure 15, we also compare ECO and NPS under the MP pricing model. We see a cost savings of 78%, which is less than that of PAYG. This is because, in the 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 an 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 1MB to 3MB; (2) Elephants Only (EO): the data usage of each S-node is uniformly distributed from 30MB to 50MB; (3) From Mice to Elephants (M-E): the data usage of each S-node is uniformly distributed from 1MB to 50MB; and (4) our default MEO.

In Figure 16, we show the costs of NPS under the aforementioned four traffic patterns in the MP pricing model (we also evaluate the overall costs of PAYG, PAYG-E, and AYCU under the four traffic patterns in Ref. [16]). 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 savings 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 Performance of the S-node Incremental Deployment Algorithm:** We compare our online S-node incremental deployment algorithm ONPS with ECO and RC. We deploy the 1,000 S-nodes in an orderly way.

In Figure 17, we present the over costs of ONPS, ECO, and RC when the pricing models are PAYG, PAYG-E, and AYCU. We can observe that ONPS outperforms RC by 67.4%, 50.9%, and 18.8% under PAYG, PAYG-E, and AYCU, respectively. We can also observe that ONPS leads a cost savings of 75.8% , 61.2%, and 93.1% compared to ECO under three pricing models, respectively. We see that when comparing the cost savings ratio between FND and ECO, the cost savings ratio between ONPS and ECO decreases. This is because FND has the overall knowledge of the deployment of all S-nodes at the beginning of each billing cycle, while ONPS gains the deployment of S-nodes in an orderly way, i.e., FND has more information of the deployment of S-nodes compared to ONPS when optimizing the deployment of N-nodes.

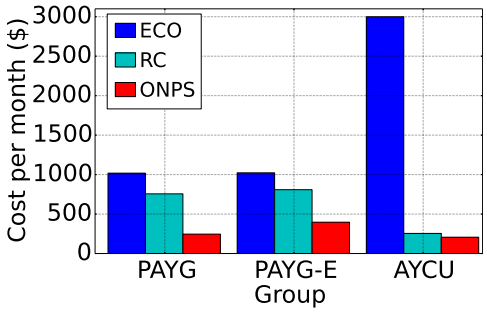


Fig. 17. The cost of ECO, RC, and ONPS under different pricing models.

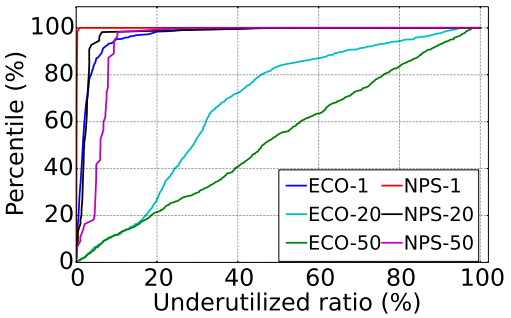


Fig. 18. The CDF of underutilized ratio of data volume.

**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 pricing models. However, to beat the cost of sTube+, ISPs have to develop pricing models with unrealistic granularity.

In Figure 18, we present the cumulative distribution function (CDF) of the ratios of underutilized data volume of N-nodes under ECO and NPS, when granularities are 1MB, 20MB, and 50MB, respectively. We can observe that the underutilized data volume ratio of all N-nodes of NPS is under 10% under the 1MB, 20MB, and 50MB 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 exceeded percentage of traffic loads of N-nodes in Figure 19. Please note that a higher exceeded percentage will lead to a worse performance, since a higher rate price is charged for each exceeding data usage unit. The subscribed data usage is not exceeded by 80% of the N-nodes of Random and 90% of the N-nodes of N-peering, 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 exceeded 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 the *over-deploy ratio*. Figure 20 shows the required over-deploy ratio as a function of $p_{req}$ under ECO and NPS. We observe that ECO needs to over-deploy many 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% of N-nodes for NPS

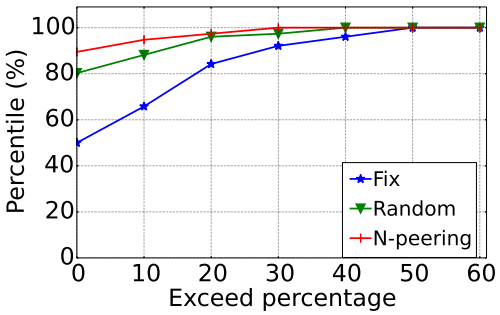


Fig. 19. The CDF of the exceed load percentage of N-nodes.

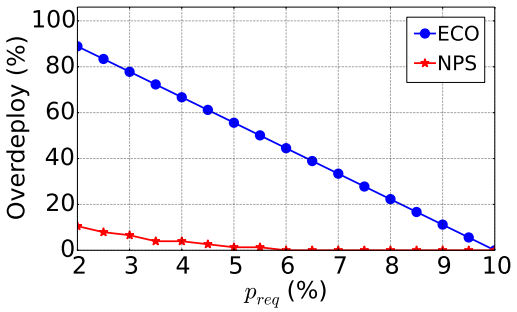


Fig. 20. The over-deploy ratio of ECO and NPS as a function of $p_{req}$.

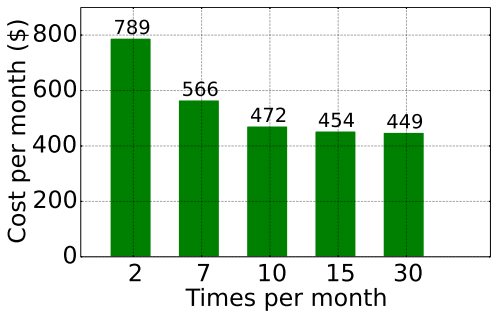


Fig. 21. The monthly cost of NPS as a function of update period.

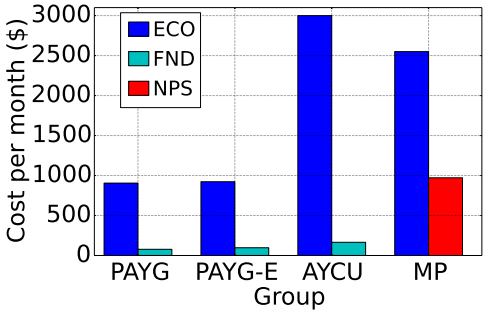


Fig. 22. The cost of ECO and sTube+ under dynamic traffic pattern.

but 89% for ECO. This illustrates that, to meet the same reliability, much fewer N-nodes should be deployed for sTube+ employing the 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 Figure 21. 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 anrb-index broadcasting that is too frequent 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 up to a 20% fluctuation.

In Figure 22, 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 Figure 22, we also compare ECO and NPS, which subscribes the data volume according to the basic monthly data volume. We see a cost savings of 62%, which is less than that of the determined traffic pattern. This is because the data volume is subscripted at the beginning of the billing cycle, and if the purchased data volume cannot cover the usage, the exceeded data volume is charged a much higher price. Under the MP pricing model, we can purchase more data volume than the basic

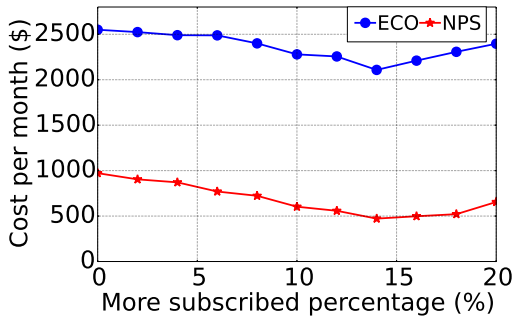


Fig. 23. The cost of ECO and sTube+ as a function of a more subscribed percentage.

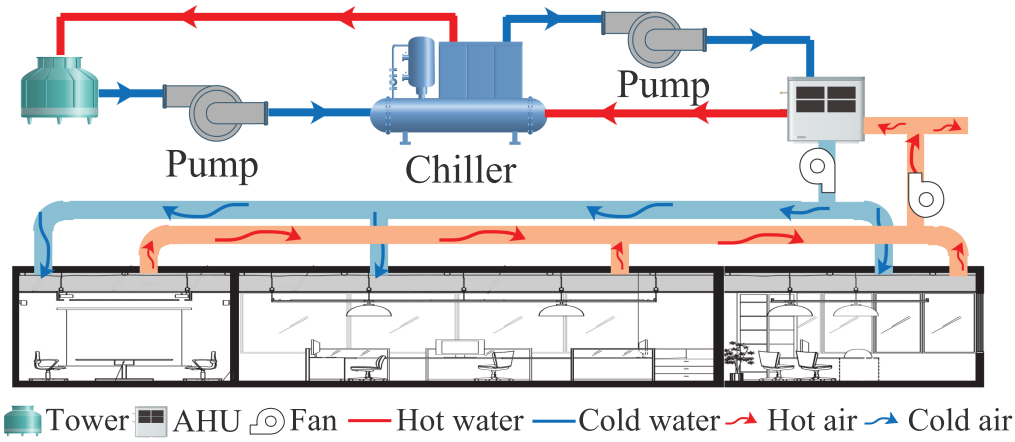


Fig. 24. A typical centralized HVAC system.

monthly data volume. In Figure 23, we show the cost of ECO and NPS as a function of a 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 lowest cost. Thus, the cost can be reduced by purchasing more data volume than the basic data volume.

## 9 A CASE STUDY

We are developing a SAMS for a centralized air-conditioning system. (Figure 24 is an illustration of a centralized air-conditioning system with water tower, chillers to cool down the water, pumps to push water circulation, an 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 COP (Equation (4)) and the performance of a pump by Water Transfer Coefficient (WTC, Equation (5)) [23, 33].

$$\text{COP} = \frac{4.181 \times Fr \times (T_r - T_s)}{W_c}. \tag{4}$$

$$\text{WTC} = \frac{Q}{W_p}. \tag{5}$$

Table 3. The Parameters for Computing COP and WTC

| 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 ($kWh$) |

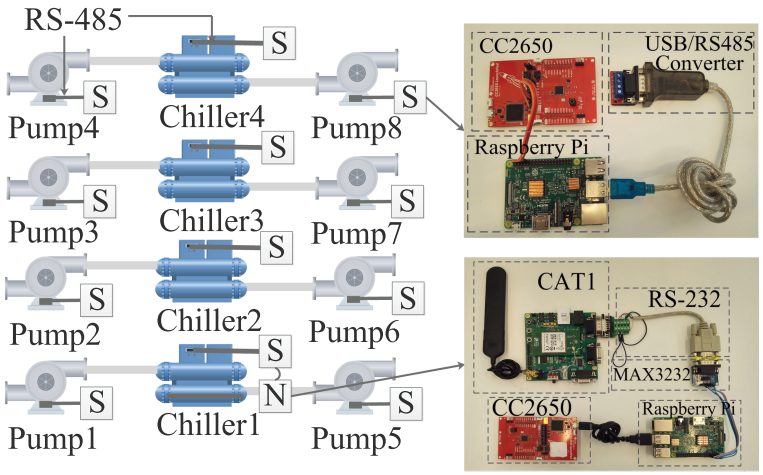


Fig. 25. SAMS supported by the sTube+ architecture.

We develop the sensing module on Raspberry Pi to collect the raw data in Table 3. Chillers and pumps have standard Application Programming Interface (API) to output data from their embedded sensors. Using a 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 [31] to query the chiller through Modbus RTU protocol. 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 Figure 25.

We deployed one N-node on a chiller and 12 S-nodes, four chillers, and eight pumps (Figure 25). All our nodes are powered by AC and we ran our system for 12 consecutive days. We rent a server deployed in Cloud Anonymity. Our system monitored the raw data successfully. We show the raw data and COPs of the chillers in Figure 26, and the raw data and WTCs of the pumps in Figure 27.[4] For better illustration, we only show chillers 1 and 2 and pumps 1 and 2.

Our cloud monitored the data consumed by each S-node (Figure 28). 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 13.76MB. The monthly communication cost of our system is $2. If adopting the ECO method, the cost is $12 which is six times more thanour method. Note that $2 is also the optimal cost we can get under the real pricing model.

[4]One-time COP/WTC is not used in practice as it is not reliable. The COP/WTC in Figures 26 and 27 is an average of 10 consecutive samples.

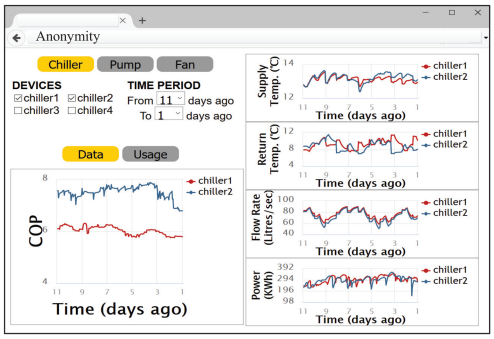


Fig. 26. The raw data and COP of chiller 1 and chiller 2.

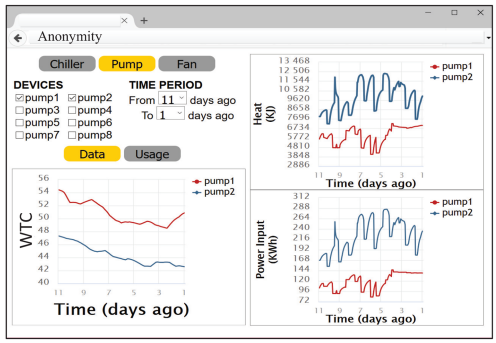


Fig. 27. The raw data and WTC of pump 1 and pump 2.

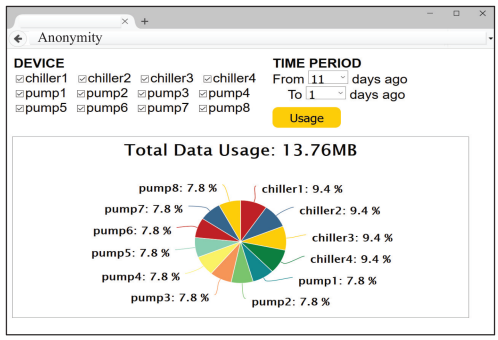


Fig. 28. The data usage of the four chillers and eight pumps.

## 10 CONCLUSION AND FUTURE WORK

One core value of the IoT 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 article 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 incremental development of devices, 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.

There are many future works. With the sTube+ data delivery architecture, there is a large space for research in price optimization for according to different pricing models and application scenarios. We also plan to develop a comprehensive SAMS for the energy systems of buildings.

## REFERENCES

[1] [n.d.]. Retrieved from https://www.singtel.com/personal/i/phones-plans/mobile/postpaid/combo/mobileshare-supplementary-plan.
[2] [n.d.]. Retrieved from https://www.o2.co.uk/sharer-plans.
[3] [n.d.]. AT&T Continues to Offer LTE-Cat 1 and LTE-Cat 4 Modules to Fit a Variety of Needs. Retrieved from https://iotdevices.att.com/uploaded_docs/lte-cat_1_and_lte-cat_4_modules_2_20170818141220039.pdf.
[4] [n.d.]. China Telecom Launches NB-IoT Service Package. Retrieved from http://en.c114.com.cn/2502/a1015477.html.

[5] [n.d.]. Deutsche Telekom launches first NB-IoT packages in Germany. Retrieved from https://www.mobileeurope.co.uk/press-wire/deutsche-telekom-launches-first-nb-iot-packages-in-germany.

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

[7] Noga Alon, Baruch Awerbuch, and Yossi Azar. 2003. The online set cover problem. In *Proc. ACM STOC'03*.

[8] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. 2009. The online set cover problem. *SIAM J. Comput.* 39, 2 (2009), 361–370.

[9] Luigi Atzori, Antonio Iera, and Giacomo Morabito. 2010. The Internet of Things: A survey. *Computer Networks* 54, 15 (2010), 2787–2805.

[10] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog computing and its role in the Internet of Things. In *Proc. ACM MCC'12*. Helsinki, Finland.

[11] Mung Chiang and Tao Zhang. 2016. Fog and IoT: An overview of research opportunities. *IEEE Internet of Things Journal* 6, 3 (2016), 854–864.

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

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

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

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

[16] Chuang Hu, Wei Bao, Dan Wang, Yi Qian, Muqiao Zheng, and Shi Wang. 2017. *sTube+: An IoT Communication Sharing Architecture for Smart After-sales Maintenance in Buildings*. Technical Report. https://goo.gl/hXEimZ.

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

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

[19] Aqeel H. Kazmi, Michael J. O'grady, Declan T. Delaney, Antonio G. Ruzzelli, and Gregory M. P. O'hare. 2014. A review of wireless-sensor-network-enabled building energy management systems. *ACM Transactions on Sensor Networks (TOSN)* 10, 4 (2014), 66.

[20] Joseph H. K. Lai, Francis W. H. Yik, and Aggie K. P. Chan. 2009. Maintenance cost of chiller plants in hong kong. *Building Services Engineering Research and Technology* 30, 1 (2009), 65–78.

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

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

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

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

[25] Antonio L. Maia Neto, Artur L. F. Souza, Italo Cunha, et al. 2016. AoT: Authentication and access control for the entire IoT device life-cycle. In *Proc. ACM Senys'16*.

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

[27] Tie Qiu, Ning Chen, Keqiu Li, Daji Qiao, and Zhangjie Fu. 2017. Heterogeneous ad hoc networks: Architectures, advances and challenges. *Ad Hoc Networks* 55 (2017), 143–152.

[28] J. S. Roessler. 2015. LTE-Advanced (3GPP Rel. 12) Technology Introduction. Retrieved from https://cdn.rohde-schwarz.com/pws/dl_downloads/dl_application/application_notes/1ma252/1MA252_2e_LTE_Rel12_technology.pdf.

[29] Petr Slavík. 1996. A tight analysis of the greedy algorithm for set cover. In *Proc. ACM STOC'96*.

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

[31] Artemios G. Voyiatzis, Konstantinos Katsigiannis, and Stavros Koubias. 2015. A modbus/TCP fuzzer for testing internetworked industrial systems. In *Proc. IEEE ETFA'15*.

[32] Thomas Watteyne, Kris Pister, Dominique Barthel, Mischa Dohler, and Isabelle Auge-Blum. 2009. Implementation of gradient routing in wireless sensor networks. In *Proc. IEEE GLOBECOM'09*. Honolulu, Hawaii.

[33] Yang Yao, Yiqiang Jiang, Shiming Deng, and Zuiliang Ma. 2004. 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 (2004), 3745–3756.

[34] Neal E. Young. 2008. Greedy set-cover algorithms. In *Encyclopedia of Algorithms*. Springer, 379–381.
[35] Thomas Zachariah, Noah Klugman, Bradford Campbell, Joshua Adkins, Neal Jackson, and Prabal Dutta. 2015. The Internet of Things has a gateway problem. In *Proc. ACM HotMobile'15*.
[36] Zimu Zheng, Dan Wang, Jian Pei, Yi Yuan, Cheng Fan, and Fu Xiao. 2016. Urban traffic prediction through the second use of inexpensive big data from buildings. In *Proc. ACM CIKM'16*.

Received January 2018; revised May 2018; accepted August 2018