The Internet of Things Still Has a Gateway Problem

Thomas Zachariah
University of California, Berkeley
Berkeley, California
tzachari@berkeley.edu

Neal Jackson
University of California, Berkeley
Berkeley, California
neal.jackson@berkeley.edu

Prabal Dutta
University of California, Berkeley
Berkeley, California
prabal@berkeley.edu

ABSTRACT

As the Internet of Things (IoT) has grown more prevalent, the gateway has become a critical linchpin of IoT network architectures. To bring constrained embedded devices online, Bluetooth Low Energy (BLE) has proven to be particularly popular for the low power sensors and actuators pervasive in the consumer IoT market. And yet, the gateways that facilitate cloud connectivity for such devices incur burdensome time, cost, and unreliability. This problem persists because current gateways conflate many application-specific functions, which continues to fuel the trend of requiring separate expensive, custom, and over-provisioned solutions for each brand or class of device to establish reliable network connectivity. One route to address the problem is the over-provisioned gateways at the heart of Apple and Google's smart home offerings. Another approach involves adding ephemeral gateway functions on pervasive connected devices like smartphones. In this paper, we explore a third approach that strips the gateway down to its bare essentials and eliminates the rest. We test the approach on the Espressif ESP32, a $3 microcontroller that contains built-in Wi-Fi and BLE radios, with a deployment of low-power IoT devices, evaluating the performance, drawbacks, and tradeoffs. Our results suggest that this is a promising technique for cost-sensitive applications with low deployment densities and aggregate data rates, but more capable design points may be preferred as these assumptions are relaxed.

CCS CONCEPTS

• **Networks** → **Network architectures**.

KEYWORDS

Internet of Things, Bluetooth Low Energy, Static, Gateway Systems, Low-Cost, Low-Powered Devices

ACM Reference Format:
Thomas Zachariah, Neal Jackson, and Prabal Dutta. 2022. The Internet of Things Still Has a Gateway Problem. In *The 23rd Annual International Workshop on Mobile Computing Systems and Applications (HotMobile '22), March 9–10, 2022, Tempe, AZ, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3508396.3512881

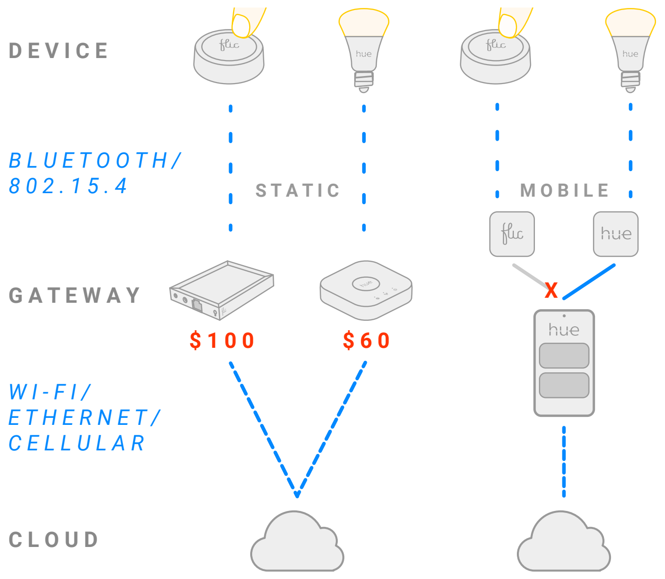


Figure 1: The Gateway Problem. Current state-of-the-art for reliable low-power IoT connectivity still typically incorporates a siloed, over-provisioned, and expensive stationary gateway for each brand or class of device. The other popular data transport mechanism is the use of device-specific apps on the owners' phones, which can opportunistically form a bridge to the Internet as a background process, but only through meager allowances of time—dictated by the OS—to receive, transmit and process data. [12, 30]

1 THE GATEWAY PROBLEM

In 2015, there was an emerging gateway problem in the Internet of Things (IoT) [35]: application-layer gateways both in software and hardware only provided application-specific connectivity to the Internet—an issue especially for constrained IoT devices, as illustrated in Figure 1. Since then, the Internet of Things has grown, permeating the consumer and industrial market sectors. There are now over 20 billion devices connected to the Internet across the globe today [21].

This number, however, is less than half of the estimate industry leaders predicted for the anticipated IoT-infused market a decade ago [5, 10]. This outcome is due, in part, to a lack of affordable, application-agnostic infrastructure through which more resource-limited devices could access the Internet. It is evident that gateways have remained the Achilles heel of low-power connectivity for the Internet of Things.

A number of factors have helped alleviate some of the past critical issues with IoT networks. For instance, alliances between several organizations across the industry have formed to agree on common standards for data protocols and user accessibility, limiting the number of custom solutions required [6, 17, 33]. Furthermore, popular cloud services and frameworks now facilitate orchestration and automation between multiple device classes, which are all assumed to have reliable network connectivity [3, 19]. The introduction of smart speakers present an additional access point to several brands of devices in the consumer space, typically limiting functionality to user-initiated controls and "scenes" [2, 4, 16].

But devices—particularly those on the edge that rely on low-power operation—still often require their own custom infrastructure or piggyback on unreliable mediums to establish connection to the Internet. Current application-specific hardware gateways are expensive one-off systems that implement different custom communication protocols for each device. The continuing standard of requiring users to obtain one for each brand of device is untenable if the IoT market is to grow.

It may be useful to leverage ubiquitous mobile infrastructure and perform gateway functions through smartphones, as has been explored in previous work [34, 35]. However, it has proven challenging to provide reliable long-term connectivity for constrained IoT devices using such methods in practice. Smartphone apps may set up a process to listen for devices and perform gateway functions for them in the background, but such operations are limited due to performance optimizations set by the phones' operating systems, especially in more recent versions of Android and iOS. This, along with the uncertainty of how often a phone with the right app will be in proximity means many devices do not receive the connectivity they need. There exists a gap in the IoT ecosystem of good quality, affordable bridging mechanisms for low-powered end devices.

We propose the use of ubiquitous, low-cost static gateways in scenarios and environments requiring long-term reliable throughput, especially with connection-less data transport for low-power devices. In pursuit of this effort, we simplify the gateway to its most essential parts — communications and processing — and we consider inexpensive off-the-shelf components that might satisfactorily work to fulfill this role. Notably the ESP32, a $3 system-on-chip (SoC) which has recently grown in popularity, incorporates Wi-Fi and Bluetooth Low Energy (BLE) radios as a convenient single package with a built-in processor and memory [11].

We explore and optimize the design and performance of gateway functions using single and multi-microcontroller implementations, particularly for reliable opportunistic data forwarding, as well as support for application needs like simple device provisioning, high-rate two-way communication, and Internet Protocol (IP) compatibility.

2 NETWORK OVERVIEW

To facilitate data transport between devices and the cloud on the simplified static gateway, we propose applying the approach from prior work [35], that specifies two general and reusable techniques for transferring data that can support many applications over a system like ESP32. An overview of the architecture is shown in Figure 2.

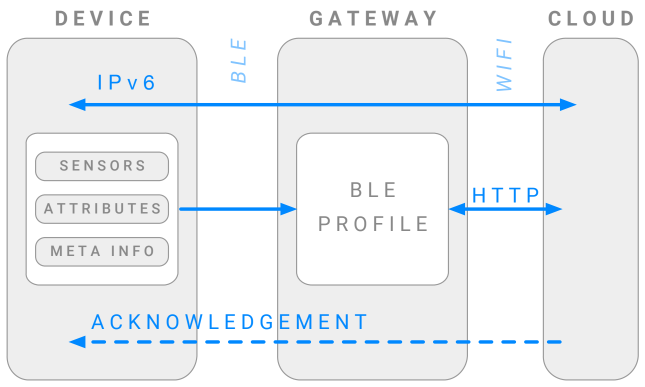


Figure 2: Architecture. We apply a previous data transport approach for mobile-based gateways to the *static* gateway problem [35]. The approach consists of two primary data transmission mechanisms: (1) via IPv6, using the gateway as an IPv6 router and treating the peripheral as an IP-connected end host, and (2) via proxy, using the gateway to forward the device's BLE profile to the cloud.

2.1 BLE Profile Proxy

In this data transport mechanism, the gateway acts as a proxy for the information contained in the BLE data structures on the peripheral. At a high level, the gateway relays the advertisements, services, characteristics, and attributes shared with it from the BLE device to a remote endpoint in the cloud.

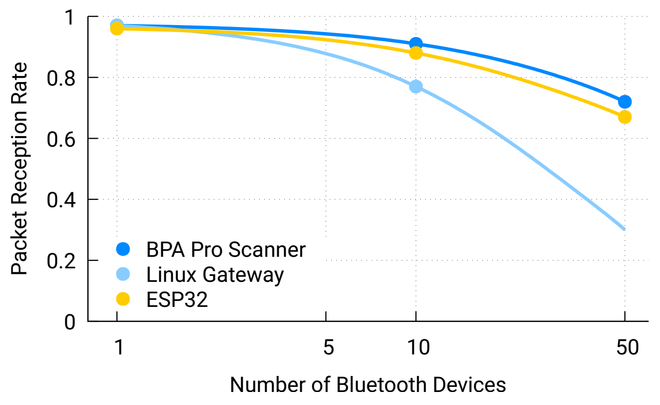
This is compatible for most existing BLE device setups, as the data organization between the peripheral and central nodes in current, application-specific BLE interaction will not fundamentally change. The approach supports opportunistic data forwarding for devices that primarily broadcast data through BLE advertisements—common for especially low power devices. Additionally, acknowledgements from the server can include commands to initiate connections and act as a proxy for two-way communication using standard BLE protocols.

2.2 End-to-End IPv6 Routing

The other data transport mechanism in our proposed network is IPv6 packet transfer over BLE or Thread 6LoWPAN. This allows each IoT device to behave as any other IP end host and take advantage of the flexibility and convenience of working at the network layer. To support this, the peripheral and gateway devices must both include a 6LoWPAN network stack.

While an official 6LoWPAN specification exists for Bluetooth [27] and implementations of the network stack are available [31], it is still not yet commonly utilized by most BLE peripherals. 6LoWPAN sees perhaps greater utilization on 802.15.4 SoCs like the Nordic nRF82540 [28] using Thread [33].

The inclusion of both a BLE and Thread-capable SoC in a gateway device allows the flexibility to support two different 6LoWPAN networks. While we consider and incorporate designs to support this, we leave the implementation and evaluation of 6LoWPAN routing to future work.



# Devices	ESP32	BPA(avg)	Linux
1	0.96052	0.96593	0.96696
10	0.88245	0.91357	0.77021
50	0.67458	0.71788	

Figure 3: BLE packet reception rates. A comparison between the PRRs of a $3 ESP32, a Linux-based gateway, and a professional $1000 Teledyne scanner (channel average) during 10 minutes of scanning with BLE devices sending unique packets every 100 ms.

3 ESP32 CHARACTERISTICS

ESP32 is an SoC from Espressif that includes Wi-Fi, BLE, two Xtensa LX6 microprocessor cores, and 520 kB internal SRAM, and can connect up to 64 MB external flash [11]. We explore the characteristics of the ESP32 to gauge its operational parameters for performing gateway operations.

3.1 Bluetooth Broadcast Packet Reception

To test the ESP32's ability to receive BLE advertisement data, we run a series of 10-minute scans in an isolated environment, with no external BLE interference. We compare with the average results of a professional $1000 Teledyne BPA scanner which runs on a single BLE channel at a time (3 total), as well as the results of a Linux-based gateway running Noble, a NodeJS Bluetooth scanning library. With devices each sending a unique packet every 100 ms, the ESP32 achieves a packet reception rate (PRR) ranging from 96% for a single device to 67% for 50, as shown in Figure 3. This performs better than the Linux-based gateway and is comparable to the average performance of the dedicated BPA scanners.

3.2 Power Draw

We take power readings using a Drok USB meter [9] at different states while running a simple gateway application that performs an active BLE scan and sends raw data via HTTP request over Wi-Fi. The readings are shown in Table 1. Most consumer Wi-Fi routers require between 4–10 W in normal operation [32]. Under the same power constraints, about 8–20 ESP32 devices could continuously run the gateway application. This number of ESP32-based gateways is more than sufficient to achieve an amount of coverage similar to that of a consumer Wi-Fi router.

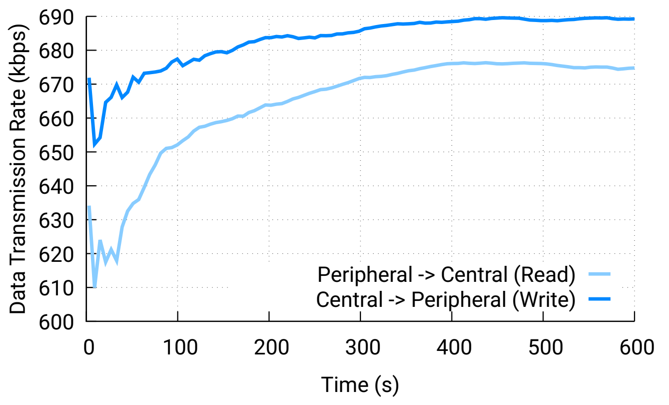


Figure 4: BLE read/write transmission rates on ESP32.

State	Power	Current
Wi-Fi & BLE off	.19 W	37 mA
BLE scanning (w/ Wi-Fi idle)	.54 W	107 mA
HTTP POST (w/ BLE scan off)	.44 W	89 mA
HTTP POST (w/ BLE scan on)	.60 W	119 mA

Table 1: Power/current at various BLE and Wi-Fi states on ESP32. Read using a Drok USB meter [9].

3.3 Bluetooth Connected Data Transport

To test read and write performance in a Bluetooth connection, we set up two ESP32s, with one in central role and the other as peripheral. Both are set to use the maximum transmission unit (MTU) of 517 bytes per read/write transaction. The central device writes data to the peripheral for 10 minutes. Then, in the 10 minutes that follow, the peripheral sends notifications to the central indicating it has data to be read. When the central receives each notification, it initiates a read.

In the test, the peripheral takes 770 ms to connect and configure itself, and the central takes 830 ms. The read/write transmission rate is plotted in Figure 4. The average transfer during read is 664 kbps, and write is 683 kbps.

3.4 Radio Coexistence

We note that the ESP32 makes a compromise to enable coexistence between its BLE and Wi-Fi radios. Both systems share the single on-board 2.4GHz radio module and antenna connection to perform their respective tasks. When both BLE and Wi-Fi are required by software, no one radio can continuously run for an extended period time. If an application runs both simultaneously, the ESP32 divvies utilization of the radio module between the two. While this still effectively facilitates connections and negotiated traffic quite reliably due to built-in delay tolerance, it proves inadequate in performing comprehensive retrieval of broadcast BLE data. When the Wi-Fi radio is running, the BLE radio only receives approximately 50% of advertisement packets with default settings. Radio priority can also be specified programatically when both radios are in use. Giving BLE priority increases PRR to around 66%.

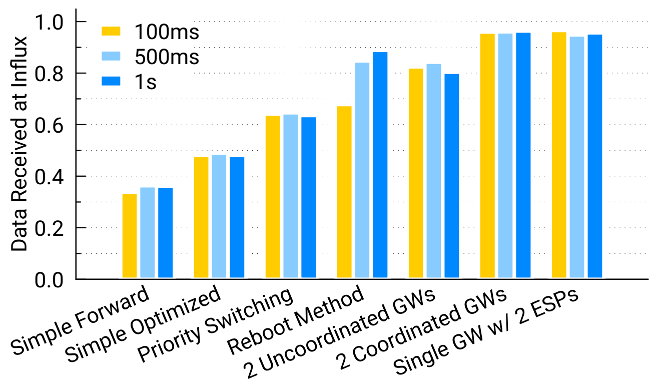


Figure 5: Forwarding performance with different approaches. Tested with unique advertisement packets sent every 100ms, 500ms, and 1s over 10 minute periods.

4 GATEWAY ANALYSIS

We test gateway operations on the ESP32 with a deployment of low-power BLE devices and evaluate its performance. As a driving application, we explore and analyze services that facilitate data transport functions for PowerBlade plug-through power meters [8]. These low-footprint devices monitor loads plugged into power outlets and use Bluetooth to relay power measurements.

4.1 Forwarding

A key indication of gateway reliability for low-power devices is performance while forwarding data from advertised BLE packets to the Internet over Wi-Fi. We test with a basic implementation that forwards advertisement data to InfluxDB, a time-series database endpoint in the cloud [20].

The devices for these tests send unique advertisement packets at intervals of 1 s, 500 ms, and 100 ms for 10 minutes each. At these advertising rates, the test device effectively simulates broadcast of the once-per-second power measurement payloads from 1, 2, and 10 PowerBlade devices respectively.

We improve forwarding using various strategies that change how the BLE/Wi-Fi coexistence ultimately impacts data reception at the cloud. The performance of each method is shown in Figure 5.

Simple Forwarding. The gateway scans for BLE packets from the the test devices and sends parsed data to Influx via HTTP POST. Without modifying default configuration values, approximately 35% of packets are received at the cloud endpoint for all of the tested advertisement intervals.

Simple Optimized. Performance improves when some of the default configuration values are modified. We increase the scan window and interval to 100ms. Data is sent to the cloud in batches of up to 160 advertisement packets. A second of delay is added after any HTTP request to allow time for other background processes to take place, and reduce failures. This increases data reception performance to nearly 50%.

Priority Switching. The gateway can only acheive 50% because of the way coexistence of Wi-Fi and BLE, and sharing of radio hardware, is handled by the ESP32, as noted in Section 3.4. Programatically switching priority of the radios as needed improves reception to about 66%.

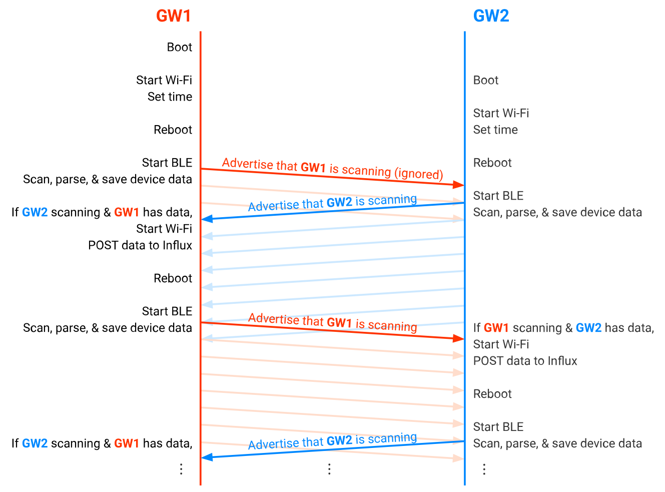

GW1
GW2
Boot
Start Wi-Fi
Set time
Reboot
Start BLE
Scan, parse, & save device data
If GW2 scanning & GW1 has data,
Start Wi-Fi
POST data to Influx
Reboot
Start BLE
Scan, parse, & save device data
If GW2 scanning & GW1 has data,
Advertise that GW1 is scanning (ignored)
Advertise that GW2 is scanning
Advertise that GW1 is scanning
Advertise that GW2 is scanning
Boot
Start Wi-Fi
Set time
Reboot
Start BLE
Scan, parse, & save device data
If GW1 scanning & GW2 has data,
Start Wi-Fi
POST data to Influx
Reboot
Start BLE
Scan, parse, & save device data

Figure 6: Timing diagram of the coordinated approach. Depicts communication between two gateways (GW1 & GW2).

Reboot Method. As we seem to reach the limit of simultaneous radio performance with the priority switching technique, we consider an approach that instead handles BLE and Wi-Fi tasks sequentially so that neither cannibalizes the other's performance while running. The ESP32 performs a BLE scan at startup. It waits until the batch limit is reached to start up Wi-Fi, connect to the network, and send data. Then it reboots to deactivate Wi-Fi, restart BLE, and repeat the process. Because ESP32 does not load a bloated OS on boot, the reset is less than half a second.

At initial glance this seems to perform relatively well, particularly for fewer advertisements. However, as the frequency of advertisements increases, the gateway struggles to support the volume due to the scanning time lost during the switch. At 100 ms, it just barely keeps up with the performance of the priority switching method.

4.2 Multiple Forwarders

Next, we explore how coverage increases when multiple gateways receive data from the same peripherals.

Uncoordinated Gateways. We test the coverage of multiple gateways, using the priority switching method, without any facilitated coordination. For two gateways situated two feet apart with the test device a foot away from both, this uncoordinated approach yields about 80% reception. This setup, however, leads to recurring periods of redundant reception when both are receiving BLE data and—more worrisome—data loss when both are in Wi-Fi mode.

Coordinated Gateways. In this approach, two ESP32s coordinate to alternately scan BLE packets from the test device and HTTP POST data to Influx. This extends from the reboot approach in which BLE tasks and Wi-Fi tasks are performed sequentially. Instead of waiting for batch limit to be reached, the BLE-scanning gateway waits for a packet from the other gateway that indicates that it has begun scanning. Once this signal is received the first gateway can halt scanning, startup the Wi-Fi radio, and send its data. It then reboots, restarts BLE scanning, and broadcasts a packet to inform the other gateway that its scanning has begun. Figure 6 depicts a timing diagram of this process. Using the same setup as before, this technique yields roughly 96% reception.

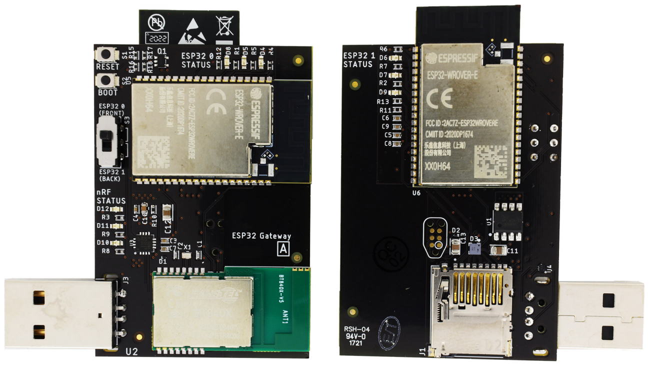


Figure 7: Gateway hardware. Our design distributes BLE, Wi-Fi, and 802.15.4 roles between two ESP32s and an nRF52840.

4.3 Connecting

For occasional scenarios requiring larger rates of data transfer from the device or communication from the cloud to the device, the gateway can facilitate high-throughput two-way transmission via BLE connection proxy. We test collection of high-fidelity readings from BLE services on PowerBlade.

The PowerBlade device has a raw sample collection service which provides a half-second sample of values (1260 points) for voltage and change of current. The device also includes a calibration service that provides the constants to adjusts the data points to known units. To retrieve a sample, the ESP32 scans for and connects to the device. Once a connection is established, the gateway reads the values from PowerBlade's calibration service. Next, it requests data from the device's raw sample service by writing '1' to the startup characteristic. The ESP32 then reads a chunk of data from the data characteristic. It repeats these request and read operations for 10 iterations. The gateway then disconnects, timestamps and adjusts the data to proper voltage and current values, and sends the data to InfluxDB via HTTP POST in two chunks. This full operation takes about 82 seconds (60 s for BLE, 22 s for Wi-Fi).

4.4 Multi-SoC Gateway

The promising results of the coordinated approach indicates that distributing BLE and Wi-Fi roles to dedicated SoCs improves performance. As a result, we explore creation of a single gateway from multiple SoCs. Table 2 compares the single- and multi-SoC setups.

Dual-ESP Gateway. This approach uses two dedicated ESP32s that communicate via SPI to operate as a single gateway. The BLE-focused ESP scans for BLE packets from peripherals. The Wi-Fi-focused ESP forwards parsed data over Wi-Fi to an Influx database. This yields a reception rate of about 96% while also avoiding scaling issues that occur in denser environments with the reboot method.

ESPxNRF Gateway. Based on these results, we design a custom high-reception gateway system, as shown in Figure 7. Using modularization to optimize performance and distribute gateway roles, the design houses two ESP32 modules, SD storage, and a Nordic nRF52840 — a supplemental chipset with Bluetooth Low Energy and 802.15.4 [28]. Using the two ESP32s for forwarding performs the same as the dual-ESP setup. Inclusion of 802.15.4 opens the door for Thread-based applications, including IPv6 connectivity [17, 33].

SoCs	Price	Approach
1x ESP32	<$5	Shared BLE & Wi-Fi
2x ESP32	<$10	Distributed BLE & Wi-Fi
2x ESP32 + 1x nRF52840	<$20	Distributed BLE, Wi-Fi, 802.15.4

Table 2: Single & multi-SoC gateway approaches.

5 DISCUSSION

In this section, we discuss lingering research questions and trade-offs that should be more deeply explored when considering use of the low-cost static gateway approach outlined in this paper.

5.1 Design

Our gateway analysis reviews a range of techniques for facilitating communication between device and cloud, each of which present a set of trade-offs. The initial forwarding technique that runs on a single ESP32 is capable of receiving roughly two-thirds of broadcast advertisements from devices within moderate range. This is likely suitable enough in more delay-tolerant deployments which only require pings at frequencies of minutes, hours, or days. The multiple forwarder and multi-SoC setups are designed to handle more frequent forwarding in more dense deployments. Our hardware design is driven by the needs of the PowerBlade deployment, which produces frequent output from devices at potentially every outlet in a household. These setups remain relatively low cost and low power even when multiple SoCs are used. Though it may be limited in resources, it runs on a simpler processing loop, and is capable of quickly recovering from crashes as it does not need to load a bloated OS every time it starts. The benefits for the low-cost static approach may fall off, however, when deployments demand highly-responsive, large-volume throughput. But such demands are often excessive for low-power IoT device deployment scenarios.

5.2 Security

When BLE devices broadcast data in advertisements as they do in the forwarding approach, that data is accessible to any scanning BLE device within range. It is important to consider this inherent risk in any deployment of BLE devices, as advertisements are a key component of the Bluetooth protocol. At minimum, sensitive data can be encrypted by the device when broadcast and translated by the trusted cloud endpoint or the gateway itself, if provisioned properly for it. Alternatively, devices can broadcast requests for the gateway to establish secure BLE connections if larger amounts of sensitive data need to be sent or if a more secure transaction needs to be facilitated via profile proxy between the device and cloud.

5.3 Industry

It is understandable why the prevailing IoT approach adopted by industry has favored expensive brand-specific gateways. At face value, it makes sense as a short-term business decision as a means for a manufacturer to generate additional revenue, guarantee buy-in by its users, and control the full pipeline between device and cloud. However, the cracks in this approach grow more apparent — made evident by the relative stagnation of consumer IoT markets and pushes for standardization between brands. The low-cost static gateway approach reduces the barrier to user entry, which reduces the barrier to device deployment, connectivity, and orchestration. One potential way to support an industry transition to this approach, is implementing a gateway-as-a-service which could run many virtual brand-specific gateways on a single physical gateway.

6 RELATED WORK

This paper re-examines and builds upon earlier claims regarding the IoT gateway problem from 2015 [35]. That work proposed leveraging mobile infrastructure to help alleviate the gateway problem, the crux of which still holds true. We find, however, that the more restrictive operating constraints of current mobile OSes and the ephemeral nature of phone presence makes them specifically less capable of fulfilling the long-term and delay-intolerant connectivity needs of many constrained IoT devices. To handle this networking gap, the work in this paper considers a *static* gateway approach—one that is also open, inexpensive, and application-agnostic.

Previous studies identify ESP32 as a desirable option for IoT applications, but focus on suitability as part of the end device [26]. While its relative low power is highlighted, its typical consumption is orders of magnitude higher than a standalone BLE SoC—the difference between lasting years on a battery instead of days [13, 22, 25]. ESP32 is, perhaps, better suited as a wall-powered IoT gateway.

For optimizing gateway performance, we look to previous work that investigates BLE IoT networks and provides strategies to set parameters for maximizing throughput [29], expand coverage with multiple gateways [14], adapt to changes in the spectral environment [18, 24], and create systems built on multi-SoC designs [15].

Examples of broader static gateway approaches exist, providing insight into the feasibility of supporitng larger scale networks, but still use a closed application-specific approach [1]. BLE mesh network techniques have been explored by academic, commercial, and standards organizations, including a strategy that builds a mesh network on top of BLE advertisements [7, 23]. A study on IPv6 over BLE explores how to establish IP-based connections with 4–8 Bluetooth devices through a central gateway system [31]. These approaches can be applied to our gateway design to provide support for wider coverage and IP compatibility.

7 CONCLUSION

The gateway remains a pain-point in current state-of-the-art IoT architectures, particularly with achieving reliable data transport for resource-constrained edge devices. We suggest an approach that anchors networking infrastructure for such systems on low-cost, pared-down open static gateways. We first test the approach on a standalone ESP32 BLE/Wi-Fi SoC, and fine-tune to reduce contention and improve performance, particularly for connectionless data forwarding scenarios in densely populated environments. For high-reliability scenarios we develop a custom gateway design which distributes gateway tasks among two ESP32 modules and an additional BLE/802.15.4 SoC. These setups can proxy as an Internet-connected BLE profile or translate to IP using 6LoWPAN. If deployed widely in requisite environments, our approach could provide cheap and reliable connectivity for a host of devices in a currently-neglected category of constrained and low-power systems, perhaps reigniting the growth of a more densely populated and useful Internet of Things.

8 ACKNOWLEDGMENTS

We wish to thank our shepherd, Feng Qian, and the anonymous reviewers for their detailed comments and feedback. This work was supported in part by the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA. This material is based upon work supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) under the award number DE-EE0008220. This document also includes work partially supported by the National Science Foundation under grant CNS-1824277, as well as by the NSF/Intel Partnership on CPS Security and Privacy under grant CNS-1822332.

REFERENCES

[1] Amazon. 2020. *Amazon Sidewalk Privacy and Security Whitepaper.* Technical Report. Amazon. https://m.media-amazon.com/images/G/01/sidewalk/final_privacy_security_whitepaper.pdf
[2] Amazon. 2021. All-new Echo (4th Gen) | With premium sound, smart home hub, and Alexa. https://www.amazon.com/All-New-Echo-4th-Gen/dp/B07XKF5RM3.
[3] Apple. 2021. HomeKit | Developing Apps and Accessories for the Home. https://developer.apple.com/homekit/.
[4] Apple. 2021. HomePod. https://www.apple.com/homepod/.
[5] Cisco. 2011. *The Internet of Things: How the Next Evolution of the Internet Is Changing Everything.* Technical Report. Cisco. https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf
[6] Connectivity Standards Alliance. 2021. Matter is the Foundation for Connected Things. https://buildwithmatter.com/.
[7] Seyed Mahdi Darroudi and Carles Gomez. 2017. Bluetooth Low Energy Mesh Networks: A Survey. *Sensors* 17, 7 (2017), 19. https://doi.org/10.3390/s17071467
[8] Samuel DeBruin, Branden Ghena, Ye-Sheng Kuo, and Prabal Dutta. 2015. PowerBlade: A Low-Profile, True-Power, Plug-Through Energy Meter. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems* (Seoul, South Korea) *(SenSys '15)*. Association for Computing Machinery, New York, NY, USA, 17–29. https://doi.org/10.1145/2809695.2809716
[9] Drok. 2019. Portable USB Doctor Multifunction Digital Meter. https://www.droking.com/usb-tester/Portable-USB-Doctor-USB-Voltmeter-Ammeter-Capacity-Meter-Energy-Meter-Temperature-Meter-Running-Time-Tester-6in1-Multifunction-Digital-Meter.
[10] Ericsson. 2010. CEO to Shareholders: 50 Billion Connections in 2020. https://www.ericsson.com/en/press-releases/2010/4/ceo-to-shareholders-50-billion-connections-2020.
[11] Espressif Systems. 2020. ESP32. https://www.espressif.com/en/products/socs/esp32.
[12] Flic. 2021. Flic | The Smart Button for Lights, Music, Smart Home and More. https://flic.io/.
[13] Mohammad Ghamari, Emma Villeneuve, Cinna Soltanpur, Javad Khangosstar, Balazs Janko, R. Simon Sherratt, and William Harwin. 2018. Detailed Examination of a Packet Collision Model for Bluetooth Low Energy Advertising Mode. *IEEE Access* 6 (2018), 46066–46073. https://doi.org/10.1109/ACCESS.2018.2866323
[14] Branden Ghena. 2020. *Investigating Low Energy Wireless Networks for the Internet of Things.* Ph.D. Dissertation. EECS Department, University of California, Berkeley. http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-209.html
[15] Branden Ghena, Jean-Luc Watson, and Prabal Dutta. 2019. Embedded OSes Must Embrace Distributed Computing. In *Proceedings of the 1st International Workshop on Next-Generation Operating Systems for Cyber-Physical Systems* (Montreal, Canada) *(NGOSCPS'19)*. Association for Computing Machinery, New York, NY, USA, 3. https://www.cse.wustl.edu/~cdgill/ngoscps2019/papers/NGOSCPS2019_Ghena_etal.pdf
[16] Google. 2021. Google Nest Smart Speakers and Displays. https://store.google.com/us/magazine/compare_nest_speakers_displays.
[17] Google. 2021. OpenThread | An Open Foundation for the Connected Home. https://openthread.io/.
[18] Albert F. Harris, Vansh Khanna, Guliz Seray Tuncay, and Robin Hillary Kravets. 2016. Smart LaBLEs: Proximity, Autoconfiguration, and a Constant Supply of Gatorade(TM). In *2016 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, IEEE, New York, NY, USA, 142–154. https://doi.org/10.1109/SEC.2016.43
[19] IFTTT. 2021. IFTTT | Do More With the Things You Love. https://ifttt.com/.
[20] InfluxData. 2020. InfluxDB Time Series Platform. https://www.influxdata.com/products/influxdb/.
[21] IoT Analytics. 2020. State of the IoT 2020: 12 billion IoT connections, surpassing non-IoT for the first time. https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/.

[22] Wha Sook Jeon, Made Harta Dwijaksara, and Dong Geun Jeong. 2017. Performance Analysis of Neighbor Discovery Process in Bluetooth Low-Energy Networks. *IEEE Transactions on Vehicular Technology* 66, 2 (2017), 1865–1871. https://doi.org/10.1109/TVT.2016.2558194

[23] Hyun-Soo Kim, Jungyub Lee, and Ju Wook Jang. 2015. BLEmesh: A Wireless Mesh Network Protocol for Bluetooth Low Energy Devices. In *2015 3rd International Conference on Future Internet of Things and Cloud*. IEEE, IEEE, New York, NY, USA, 558–563. https://doi.org/10.1109/FiCloud.2015.21

[24] Jia Liu, Canfeng Chen, Yan Ma, and Ying Xu. 2013. Adaptive Device Discovery in Bluetooth Low Energy Networks. In *2013 IEEE 77th Vehicular Technology Conference (VTC Spring)*. IEEE, IEEE, New York, NY, USA, 1–5. https://doi.org/10.1109/VTCSpring.2013.6691855

[25] Jia Liu, Canfeng Chen, Yan Ma, and Ying Xu. 2013. Energy Analysis of Device Discovery for Bluetooth Low Energy. In *2013 IEEE 78th Vehicular Technology Conference (VTC Fall)*. IEEE, IEEE, New York, NY, USA, 1–5. https://doi.org/10.1109/VTCFall.2013.6692181

[26] Alexander Maier, Andrew Sharp, and Yuriy Vagapov. 2017. Comparative Analysis and Practical Implementation of the ESP32 Microcontroller Module for the Internet of Things. In *2017 Internet Technologies and Applications (ITA)*. IEEE, IEEE, New York, NY, USA, 143–148. https://doi.org/10.1109/ITECHA.2017.8101926

[27] Johanna Nieminen, Teemu Savolainen, Markus Isomaki, Basavaraj Patil, Zach Shelby, and Carles Gomez. 2015. IPv6 over BLUETOOTH(R) Low Energy. RFC 7668. https://doi.org/10.17487/RFC7668

[28] Nordic Semiconductor. 2019. nRF52840 - Bluetooth 5.2 SoC. https://www.nordicsemi.com/Products/nRF52840.

[29] David Pérez-Diaz de Cerio, Ángela Hernández, Jose Luis Valenzuela, and Antonio Valdovinos. 2017. Analytical and experimental performance evaluation of BLE neighbor discovery process including non-idealities of real chipsets. *Sensors* 17, 3 (2017), 499. https://doi.org/10.3390/s17030499

[30] Philips. 2021. Smart Lighting | Hue. https://www.philips-hue.com/.

[31] Michael Spörk, Carlo Alberto Boano, Marco Zimmerling, and Kay Römer. 2017. BLEach: Exploiting the Full Potential of IPv6 over BLE in Constrained Embedded IoT Devices. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems* (Delft, Netherlands) *(SenSys '17)*. Association for Computing Machinery, New York, NY, USA, Article 2, 14 pages. https://doi.org/10.1145/3131672.3131687

[32] The Power Consumption Database. 2022. The Power Consumption Database. http://www.tpcdb.com/list.php?type=11.

[33] Thread Group. 2021. Thread. https://www.threadgroup.org/.

[34] Thomas Zachariah, Joshua Adkins, and Prabal Dutta. 2020. Browsing the Web of Connectable Things. In *Proceedings of the 2020 International Conference on Embedded Wireless Systems and Networks on Proceedings of the 2020 International Conference on Embedded Wireless Systems and Networks* (Lyon, France) *(EWSN '20)*. Junction Publishing, USA, 49–60.

[35] Thomas Zachariah, Noah Klugman, Bradford Campbell, Joshua Adkins, Neal Jackson, and Prabal Dutta. 2015. The Internet of Things Has a Gateway Problem. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications* (Santa Fe, New Mexico, USA) *(HotMobile '15)*. Association for Computing Machinery, New York, NY, USA, 27–32. https://doi.org/10.1145/2699343.2699344