Freeloader's Guide Through the Google Galaxy

Joshua Adkins
University of California, Berkeley
Berkeley, California
adkins@berkeley.edu

Branden Ghena
University of California, Berkeley
Berkeley, California
brghena@berkeley.edu

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

ABSTRACT

One of the largest impediments to pervasive sensing is ensuring equally pervasive network access. While we can create wireless sensors that last for years without human intervention, the network infrastructure to support their deployment requires planning, power, and maintenance. The potential for a crowd-based solution to this problem is ripe—ever pervasive smart phones have the hardware and connectivity to serve as ubiquitous mobile gateways—however the fragmentation of low power wireless protocols combined with the lack of incentive for users to sacrifice their own resources transporting others' data has made this approach untenable.

Through an "off label" use of Google's Physical Web and Nearby Notifications, it was possible to ignore these problems and exploit nearly the entire global population of Android phones to slowly transport sensor data to an arbitrary web server. This mechanism was enabled by default and transparent to the phone's user. On one hand, it served as an exciting opportunity to explore infrastructure-free wireless networking. In a one week deployment of five devices transmitting at 1 Hz, we were able to successfully transport 326 kB of data with an average data rate of 0.1–2.6 bps. This is slow, but sufficient for many applications such as environmental monitoring and sensor status reporting. On the other hand, a mobile operating system probably should not have enabled exfiltration of arbitrary data without a user's knowledge or consent. While Nearby Notifications has now been decommissioned, we examine security policy requirements for future systems that interact with nearby devices, and we envision a similar, intentional mechanism to allow data hitchhiking for the Internet of Things.

CCS CONCEPTS

• **Networks → Network experimentation**; *Mobile and wireless security*; *Mobile ad hoc networks.*

KEYWORDS

Physical Web, Gateway, Android, Data transport

ACM Reference Format:
Joshua Adkins, Branden Ghena, and Prabal Dutta. 2019. Freeloader's Guide Through the Google Galaxy. In *The 20th International Workshop on Mobile Computing Systems and Applications (HotMobile '19), February 27–28, 2019, Santa Cruz, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3301293.3302376

1 INTRODUCTION

Google deployed the largest standardized and widely available network for freely transporting extremely low-rate data from resource-constrained devices, realizing coverage of as much as 20% of the land area of the globe [10]. Interestingly, it is not clear whether they are aware of their accomplishment. The Nearby notification service [9], part of Google's Physical Web [6], aimed to facilitate interactions with physically close devices. However, it turns out that this same service, installed and automatically running on every Android smartphone between June 2016 and December 2018, could be used to ferry sensor data.

The desire for ubiquitous, seamless connectivity for sensor networks has remained unfulfilled for some time now. The research and industrial community has long explored a plethora of techniques for enabling low-cost data transport for deployed sensors, but while there are numerous MAC protocols and standards targeting resource-constrained devices, the reality is that no single leading standard has emerged. On top of this, reliable network deployments require system administrators to install and maintain gateways to provide coverage, an infrastructure requirement adding unwanted overhead to real-world applications.

One enticing solution to this infrastructure challenge is the smartphone, and there have been several proposals suggesting their use as gateways for pervasive sensors [5, 15, 18]. Phones go everywhere people go, contain low-power networking hardware, maintain near constant connectivity to the cloud through cellular infrastructure, and have human oversight to maintain their functionality. Unfortunately, we have never figured out how smartphone gateways would work in practice. Just using smartphones does not fix the problems with standardization, and users are hesitant to waste their precious battery and data limits to transport others' data. Furthermore, there is an underlying concern of transporting malicious data or leaking private information by acting as a gateway.

While these concerns have rightfully slowed the explicit adoption of personal smartphones for generic data transport, we leverage what was arguably an architectural bug in Google's Physical Web to explore the possibility of ubiquitous smartphone data transport. The mechanism relied on Nearby Notifications, a service which transported URL data contained in Bluetooth Low Energy (BLE) [2] beacons through HTTP GET requests to populate summary information about each device. Until December 2018, Nearby ran by default on nearly all Android phones with Bluetooth enabled. The amount of data transported was small, only 10 bytes at a time; and slow, only transmitting data when a phone transitioned from idle to active. But it resulted in a pervasive mechanism by which sensors could send arbitrary data through smartphones carried by over half of the population. This possibility allows us to explore 1) how well this data channel worked in practice, 2) the security implications of its existence, and 3) the opportunity to use smartphones as universal gateways moving forward.

In a one-week deployment of five devices around UC Berkeley's campus, we were able to transmit a total of 326 kB of data with an average data rate ranging from 0.1–2.6 bps. The channel had very low packet reception rates (PRR) ranging from 0 to 16%, and equivalently poor energy efficiency, but we were still able to receive data from sensors without any network deployment or management outside of a web server that was created to receive the HTTP GET requests. This transport mechanism would not have been a good choice for applications which need reliable, timely, or data intensive transport, but it may have been perfectly suited to many other applications such as low-rate environmental sensing, collecting metrics about the usage of buildings or trails, and reporting the status of sensors which are not currently networked.

In general, the only way to detect that a device was using your phone for data transport was by actively scanning for BLE packets that meet the Physical Web beacon specification. Furthermore, this service was enabled by default on Android phones, and the only way to stop your phone from being used was to turn off Bluetooth or explicitly disable the Nearby notification service. If viewed as a side channel for data exfiltration, this mechanism offers plenty of bandwidth to offload a key or password, and it would be particularly applicable for this use case given that a malformed BLE packet could be transported without any notification to the user. While architectural changes to the Physical Web could have significantly slowed down this method of data transport, and the notification service should not have performed HTTP GET requests without displaying them to the phone's user, it would be difficult to entirely stop the leakage of information from a beaconing sensor to its associated web server and still achieve the original goals of the Nearby notification service.

Between the original writing of this paper and its publication, Google announced the shutdown of Nearby Notifications due to the increasingly spammy nature of messages being received by users [1], and on December 6, 2018, the service was discontinued. Therefore, the transport mechanism reported in this paper is no longer functional. We do not believe, however, that this nullifies the findings of the work. Google still deployed the largest ad-hoc network for resource-constrained devices to date. Using this network was incredibly easy because it required only knowledge of BLE and HTTP, ubiquitous technologies in sensor networks and the web respectively. Finally, we were able to successfully transport a relatively large amount of data without over-utilizing the resources of any one phone individually. Looking forward, this leads us to imagine a world in which a data transport mechanism like this one is encouraged and nurtured, and explore solutions to the economic, standardization, and privacy concerns that might otherwise prevent a wide-scale crowd-based data backhaul.

2 BACKGROUND & RELATED WORK

First, we explore the background that lead to the Physical Web as it exists today. Then we explore the works that inspire and are benefited from opportunistic smartphone data transfer.

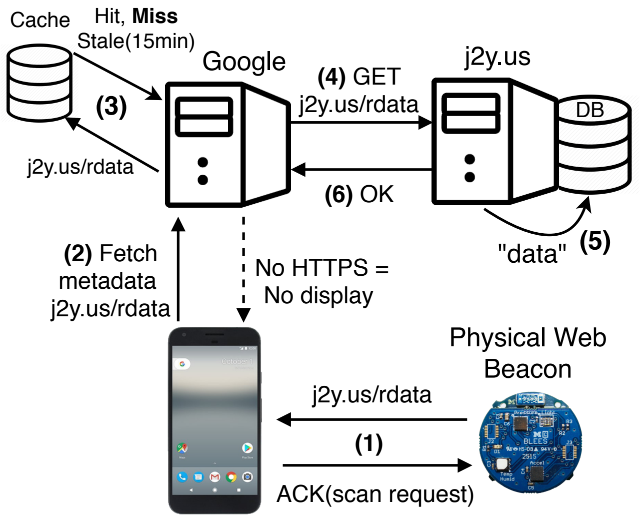


Figure 1: Architecture of Google's Physical Web and our transport mechanism. (1) In the Physical Web smartphones scan for Eddystone-compliant beacons using BLE and respond with a scan request when a beacon is received. **(2)** The phone requests that a Google server fetch metadata about the URL in that beacon. **(3)** If the server has the metadata for that URL cached, it will respond from its cache, **(4)** otherwise it will GET the URL from the remote server. **(5)** If that server is under control of a third-party, they can store the URL path as data and **(6)** respond with an OK. If the beacon and server do not use HTTPS none of this is displayed on a user's phone. To prevent caching of repeat data, some randomness 'r' can be appended into the URL path. This process was enabled by default on phones running Android 4.4 and newer through the Nearby notification service and occurs for a few seconds every time the phone transitions from idle to active.

Eddystone, Physical Web, and Nearby. For the last four years Apple and Google have been pushing to make the distributed set of devices and sensors that comprise the Internet of Things more scalable and interactive, and they have attempted to accomplish this goal by binding physical devices to web-based resources. In 2014, Google released the UriBeacon protocol, which enables BLE beacons to point to URLs [7]. UriBeacon was replaced by the Eddystone standard, which adds the ability to beacon unique IDs and telemetry data in addition to URLs [8]. In an Eddystone packet, which is what we use for this work, URLs can be up to 17 bytes, not including the protocol specification (e.g., http://).

The Physical Web project is a browser for physical devices which uses Eddystone beacons (along with other discovery protocols) to find and display relevant web content [6]. The architecture for the Physical Web is shown in Figure 1. In the Physical Web, the smartphone listens for Eddystone URL beacons, the contents of which are fetched from a Google server when the user requests. If the page is in the server's cache, it is sent to the user, otherwise it is fetched from the remote location then served to the smartphone and cached on the server for some period of time.

Nearby Notifications is a service which by default ran in the background on all phones running Android 4.4 or newer [9] starting in June 2016. If the user's Bluetooth was turned on, and Nearby was not manually disabled, it would perform a BLE scan every time the phone transitioned from an idle to active state (such as when the user wakes up their screen). Upon receiving an Eddystone beacon, it would fetch the metadata for the advertised URL using an HTTP GET request, displaying a notification which the user could click to open the website. If the Eddystone beacon did not conform to the Physical Web and Nearby security standards, the metadata would still be fetched from the remote server, but the beacon would not be displayed to the user. Other phones, such as iPhones, could still receive and use Eddystone beacons to find nearby devices, but they did not run Nearby Notifications.

In October, 2018 Google announced that it would shut down the Nearby Notification service due to spammy notifications appearing on users' phones, and in December, 2018 the service was stopped [1]. This means that notifications are no longer being pushed to users' smartphones about nearby devices and the fetching of metadata about nearby devices is no longer operating as a background service.

Mobile Phones as Gateways. The ubiquity of smartphones makes them a prime target for providing opportunistic connectivity to devices. Fürst et al. investigate the use of smartphones as access points for home appliances [5], guaranteeing that a user is present when communication occurs and alleviating some security concerns. Classic Bluetooth has been used to provide communication for devices in oilfields and offices [15]. The authors find that intentional visitation of sensors is often required given the limited range of Bluetooth. While some applications exist, a large area of research remains in determining what services smartphone gateways should provide to sensors, in creating policies for handling data flows, and in providing incentives to users that are providing this access [18]. Furthermore, little research has performed any at-scale evaluation of smartphones as gateways.

Applications of Ubiquitous Intermittent Networking. For low power sensors, getting internet connectivity is a difficult process. Maybe they are deployed along hiking trails with no WiFi to be found for miles [11, 17]. Maybe they are deployed throughout cities [3, 16], where networks are everywhere, but none of them are available for transporting your data. All of these locations are short on low-power connectivity but high on foot traffic. Data transfer through Physical Web beacons could breathe life into these applications by ubiquitously and cheaply communicating measurements. These ideas also connect strongly to delay tolerant networking [4]. While beacons may be unattended for long stretches, the eventual presence of a smartphone enables application data to be transmitted. While the explored networking channel is not delay tolerant, it may be possible to include delay tolerance as a feature of future smartphone gateways.

3 HOW DATA CAN HITCHHIKE

To transfer data through the Physical Web, we encode it into the Eddystone beacon URL. When an Android user wakes up their phone, the phone receives these beacons and requests the metadata for that URL from a Google server. If the metadata associated with that URL is not already cached, the server calls HTTP GET on the URL. The remote server can then extract data encoded in the path of the requested URL and respond to the request with arbitrary data. This process is shown in Figure 1. Notably, if the URL specified is not using the HTTPS protocol, then the metadata for this beacon/URL will not be displayed on the user's phone as it would have been for a valid Nearby Notification.

Therefore, to receive data through this mechanism all you need to do is set up a web server with a short domain name and record the paths of each request to this web server. Eddystone beacons can contain up to 17 bytes of URL data, and the shortest domain name we could reasonably purchase was j2y.us, leaving every beacon up to 11 bytes of payload.

Cache Prevention. To ensure that every beacon has the opportunity to hit our server, we must prevent Google's server from caching the URL. We were unable to find any way to do this from the server side, so instead we introduce a nonce to the URL.

To calculate the minimum length of this nonce, we first need to establish the cache refresh rate of Google's servers, as demonstrated in Figure 2. In the experiment we find that the inter-request arrival time for beacons sending the same URL is around 15 minutes, and confirm that it is significantly lower for URLs that are not repeated. At a 1 Hz transmission rate and a nonce overflow of 30 minutes to be conservative, 11 bits of nonce is required to prevent caching of repeated data. Subtracting this from our available payload, each beacon is capable of transmitting a little more than 9 bytes of data.

Acknowledgements and Adaptation. One of the largest downsides to this data channel is that it is unidirectional. Because data transport is largely based on foot traffic, this means that beacons spend significant periods of time transmitting with no smartphone present to transport their data.

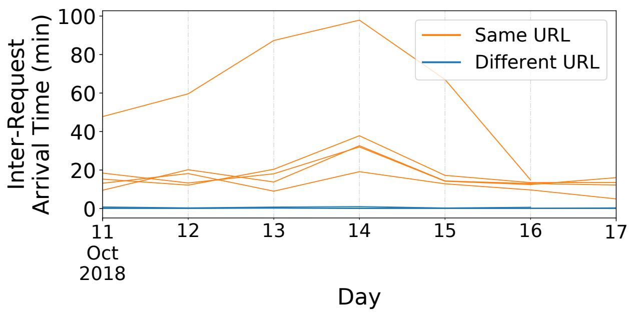


Figure 2: The daily average inter-request arrival time in minutes for devices beaconing the same URL vs rotating URLs. From this plot we see that nearly all beacons from rotating URLS result in requests to our server and we can estimate the amount of time until Google's servers considered a cached URL stale and fetched the page again from the remote server. We estimate this time to average around 15 minutes, but as high as 30 minutes on some days such as October 14th. We believe the beacon with very high inter-request arrival times to be an outlier due to a combination of caching and very low reception rate, not indicative of the actual cache refresh rate. To ensure repeated data gets to the remote server, beacons transmitting at 1 Hz with a 30 minute cache refresh rate, must include an 11 bit nonce with their data.

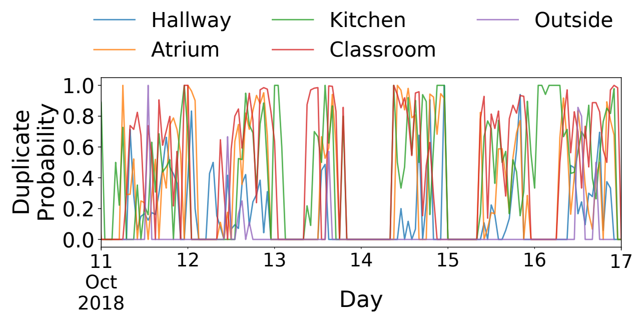


Figure 3: The probability of the same packet being received more than once by the server, every hour. High duplication probabilities indicate that the network could carry more capacity, and that beacons could take advantage of that capacity by transmitting faster than our 1 Hz transmission rate. Transmitting faster should lower the probability that two phones receive the same beacon. For our packet reception rate and throughput calculations we deduplicate the data, underestimating the maximum throughput.

To alleviate this, we propose using the "scan request" mechanism of BLE. Scan requests are responses to beacons that scanners transmit to request additional information from the device. While searching for Physical Web devices, Android phones automatically transmit a scan request for each beacon they discover. Beacons can use this scan request to accomplish two goals: 1) have some confidence that transmitted data will reach the remote server, and 2) adapt their transmission rate to better reflect the current phone density in an attempt to save energy.

Unfortunately, because scan requests from multiple scanners often conflict [12], waiting for a scan request to confirm data transmission may significantly reduce a beacon's data rate. To save energy, nodes could back off their transmission rate the longer they go without a scan request, then beacon very quickly on the first scan request they receive. For example, in an indoor, commercial setting, as long as the beacon does not slow its transmission so much they that it fails to detect a scan request early in the morning, this would allow it to save energy throughout the night and have higher data throughput during the day when phones are present.

4 EXPERIMENTAL RESULTS

To evaluate the channel capacity and packet reception rate we deploy 10 beacons around UC Berkeley's campus for one week. Deployments occurred in pairs, where one beacon in each pair is transmitting the same URL, and the other is transmitting an incrementing URL. Both of these beacons transmit at 1 Hz, and we log all requests received by the web server. Data analysis is only performed on the five beacons with incrementing URLs because the transmissions from the other beacons are often cached as explained in Section 3. Nodes are deployed in variety of locations, including a kitchenette, a heavily-used classroom, a hallway between offices, a small atrium, and an outdoor garden near a sparsely traveled path.

Duplicate packets. Because each incrementing URL is only transmitted once, we did not expect to receive a significant number of duplicate packets. Upon analysis we find that during periods of high foot traffic there is a high probability that packets are duplicated. This was most likely caused by two phones receiving a packet at the same time and triggering a request from a google server before the URL from either request could be cached. This is effectively wasted network capacity; if the beacons were transmitting faster, then the probability that two phones receive the same beacon should be lower. The probability of duplicate packets over time is shown in Figure 3. We see that there are periods where nearly all packets are duplicated at least once, and often packets are duplicated multiple times. We would certainly see higher total throughput if we increased transmission rate, but because it is difficult to quantify the degree of increase, we remove all duplicate packets in the remainder of our analysis.

Packet reception rate. To measure the packet reception rate, we deduplicate the received packets as mentioned above. In Figure 4, we see packet reception rates that correlate with times that the respective spaces are occupied. The burstiness of packet reception rate indicates that back-off mechanisms based on scan requests could be highly effective, however we did not evaluate the impact of this policy experimentally.

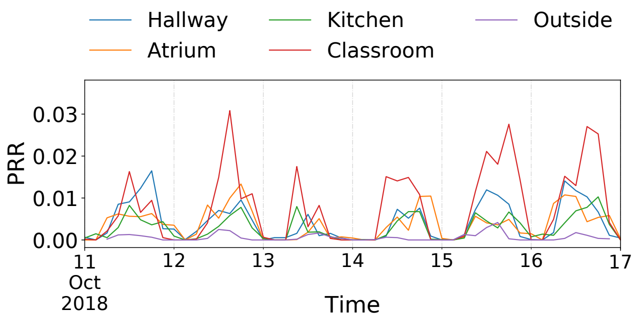
Throughput. Because every packet carries the same amount of data and beacons are transmitting at a uniform rate, network throughput is a scalar multiple of the packet reception rate. We find average network throughput to range from 0.1 bps for the outdoor node to 2.6 bps for the heavily trafficked classroom, with other nodes averaging around 0.5 bps. This throughput does not reflect the reliable network throughput. In many applications, data redundancy or the scan request acknowledgment mechanism discussed in Section 3 would have to be introduced to ensure data reception, and this would reduce final data throughput for a network of this type.

5 DISCUSSION

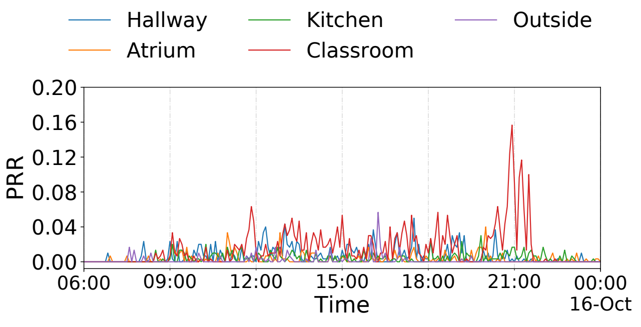
Never before have we been able to crowd-source data transport through such a large percentage of the population. The idea that we could stick a random sensor nearly anywhere and eventually receive data from it is unprecedented. At the same time, the use of this service did present a security vulnerability, and similar vulnerabilities could easily be present in future services that interact with nearby devices, especially if they are enabled on a large number of smartphones by default. We therefore use this section both as an opportunity to examine potential lessons learned about security from this experiment, and to imagine a service such as this one with the necessary oversight and capabilities such that the benefits of universal data transport outweigh the security risks.

5.1 Safely Interacting with Nearby Devices

With the Nearby notification service now discontinued, we do not feel it necessary to explore the specific methods by which Google may fix the service, however we do believe analyzing the primary reasons why this could be used for data exfiltration is useful for future services interacting with proximal devices. At its core the Nearby notification service could be used for data exfiltration because 1) it was running en masse on smartphones without the knowledge of many users and 2) it was interacting with devices without clearly notifying the user of these interactions.



(a) Full Week PRR



(b) Single Day PRR

Figure 4: (a) The packet reception rate (PRR) plotted across a full week averaged over 3 hour intervals, and (b) the PRR plotted for a single day averaged over 5 minute intervals. From the PRR we can see that, as we expect, more highly trafficked areas successfully transmit more packets. In (a), the highest PRR is in the classroom during a homework/office hours session that fills up the room, and we see the sparsely traveled walkway consistently has the lowest PRR. While these are low packet reception rates, falling to zero during the middle of the night, all nodes successfully transmitted some packets each day, even during the weekend.

Even with the brief Bluetooth listening window on each phone and the caching implemented to limit the number of requests received by each device's web server, the sheer scale of all Android devices performing a specific task is powerful enough to result in significant data transport. This leads us to believe that Google should have used much stronger caching policies when implementing Nearby Notifications and more generally that the side effects of background services operating at this scale need to be more carefully considered. Other research has discussed the need to make peripheral interactions more apparent to a device's user [13, 14], and we believe this argument should extend to external devices, especially for interactions not explicitly initiated by the user.

On a more positive note, this experiment exposes the benefits of MAC address randomization. We at first hypothesized that this mechanism could have been used for low-infrastructure people-tracking by recording the MAC addresses of scan requests and then sending them through that same persons's smartphone. However, the BLE MAC address randomization present in newer versions of Android, rotates addresses quickly enough that this is not possible. It may have still been possible to scan for more permanent MAC addresses on another networking interfaces (such as the MAC addresses associated with a WiFi connection) and transport them over the proposed mechanism, but this would greatly increase the complexity of the system.

5.2 Envisioning a Crowd-Sourced Gateway

An important path forward is to address the core problems holding back the vision of a global, crowd-sourced gateway: compensation for used resources, security and privacy concerns, and a minimal but sufficient set of services for universal data transport.

Compensation. To compensate for used resources we imagine a global clearinghouse for transferred data. Data is shuttled from a user's phone, to their mobile service provider, then from their mobile service provider to the clearinghouse. To retrieve the data, entities must both verify their identity (most likely using existing PKI) and pay the going rate for data transfer, which can then be applied to directly to the bill of the user who transferred the data. While rather complicated, a solution like this one could be almost entirely facilitated a large company like Google, Samsung, or Verizon. Also important, users need to be able to set limits as to how much data they transfer, and how much battery this can consume. Given the utility of even small amounts of sensor data, we do not expect either of these be technically limiting so much as necessary for providing users with the agency to enforce security and privacy constraints. Additionally, the amount of data transferred by these devices is small relative to the image and video heavy mobile browsing users do on a daily basis. Even if all 326 kB collected during this study had been transferred by a single phone, that user's costs would have been affected imperceptably.

Security and Privacy. If users are transferring others' data, there will always be a chance for data exfiltration. Solutions to prevent it border on contradictory to the goal, however allowing users to audit the list of owners or domains for the data they have transferred is at least a step in the right direction. We also imagine encouraging users to create geofenced zones in which data should not be transported except from approved devices to prevent easy tracking of users in and around their homes. Secure facilities and corporate campuses would need to continue down the path of both active network scanning to eliminate threats and policies to turn off network interfaces or remove phones entirely. From a privacy perspective, any global service, through Google or otherwise, should continue operating as a proxy for requests so that owners of beacons can not identify the IP address or MAC address of a courier phone.

Architecture of Smartphone Gateway. A universal smartphone gateway could provide a full and capable networking solution for resource constrained sensors including reliable and bidirectional data transport, delay tolerance, time updates, and even phone-collected information such as location to help contextualize the data being transported. We explore the possibilities for these extra services in our prior work [18]. However, one of the largest impediments to realizing these services in practice is standardization, and the reason that this transport mechanism was so easy to implement was its use of well-understood interfaces such as BLE advertising and HTTP to transport the data. Even though this architecture only provided unreliable, unidirectional communication, it still can support a sufficient number of applications that a more intentional version may gain traction given a proper compensation scheme and transparency policy.

Moving forward, the use of a specific scan request packet or a BLE connection could be formalized to acknowledge successful custody transfer, and delay tolerance could be supported by simply waiting to forward data until a cellular signal is available. These two additions would add significant functionality and enable applications such as off-the-grid environmental monitoring without adding significant complexity. To enable bidirectional communication and end-to-end acknowledgment of data transport, we imagine an HTTP proxy service which utilizes BLE connections to send HTTP requests and return HTTP responses similar to that proposed in our prior work [18]. We again stress, however, that we believe the expansion of gateway services to support all or even a large portion of IoT applications is likely to make standardizing these services more difficult. Further, applications which need reliable or bidirectional communication are much more likely to find the opportunistic nature of a smartphone gateway lacking.

5.3 Ethics

As a final aside, we would like to briefly discuss our ethics in running this experiment. To start, we collected no human data or personally identifiable information; all requests to our server came directly from a Google server. Because we do not have identifiable human subjects, we are outside the scope of an IRB. We also note that the potential harms of our experiment are incredibly low, potentially non-existent. All Android phones were already running this feature, and there are already many beacons deployed around campus, so the traffic generated by our beacons is a fraction of the existing beacon traffic and should not impact battery life or data use of nearby phones. Finally, the ability of any entity to immediately use this feature upon submission was limited because it requires physically deploying new beacons or performing a firmware update on existing beacons, neither of which seem plausible at scale. Of course now the ability of someone to use this service maliciously does not exist due to its discontinuation.

6 CONCLUSIONS

The Physcial Web is envisioned and intentioned to support nearby intelligent spaces, bridging the gap between users and the devices around them. In practice, the Physical Web and Nearby Notifications was also the single largest network backhaul for low-power devices ever deployed, literally affording coverage everywhere people go. Now that Nearby Notifications has been discontinued, where do we go from here? We could view this as a now-patched security vulnerability, for both persons and property owners, noting that unwitting individuals may have been inadvertently supporting the exfilitration of data. Or, we could could choose to recognize this as a missed opportunity. By characterizing the impact of the current system and adding just enough capability for a wider set of applications, we could finally enable the dream of pervasive, low-power data backhaul.

7 ACKNOWLEDGMENTS

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. Additionally, this material is based upon work supported by the National Science Foundation under grant numbers CNS-1824277, DGE-1256260, and DGE-1106400, and NSF/Intel CPS Security under grant CNS-1822332.

REFERENCES

[1] Android Developers Blog. 2018. Discontinuing support for Android Nearby Notifications. https://android-developers.googleblog.com/2018/10/discontinuing-support-for-android.html. (25 Oct 2018).

[2] Bluetooth Special Interest Group. 2018. Bluetooth SIG. https://www.bluetooth.com/. (Mar 2018).

[3] Yun Cheng, Xiucheng Li, Zhijun Li, Shouxu Jiang, Yilong Li, Ji Jia, and Xiaofan Jiang. 2014. AirCloud: a cloud-based air-quality monitoring system for everyone. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems (SenSys'14)*. ACM.

[4] Kevin Fall. 2003. A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM'03)*. ACM.

[5] Jonathan Fürst, Kaifei Chen, Mohammed Aljarrah, and Philippe Bonnet. 2016. Leveraging physical locality to integrate smart appliances in non-residential buildings with ultrasound and Bluetooth Low Energy. In *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI'16)*. IEEE.

[6] Google. 2017. The Physical Web. https://google.github.io/physical-web/. (Jun 2017).

[7] Google Inc. 2015. UriBeacon Open Source Project. https://github.com/google/uribeacon. (July 2015).

[8] Google Inc. 2017. Eddystone. https://github.com/google/eddystone. (Apr 2017).

[9] Google Inc. 2018. Nearby. https://developers.google.com/nearby/. (Oct 2018).

[10] GSM Association. 2012. Universal Access – How Mobile can Bring Communications to All. https://www.gsma.com/publicpolicy/wp-content/uploads/2012/03/universalaccessfullreport.pdf. (2012).

[11] Jyh-How Huang, Saqib Amjad, and Shivakant Mishra. 2005. Cenwits: a sensor-based loosely coupled search and rescue system using witnesses. In *Proceedings of the 3rd international conference on Embedded networked sensor systems (SenSys'05)*. ACM.

[12] Robin Kravets, Albert F Harris, III, and Roy Want. 2016. Beacon trains: blazing a trail through dense BLE environments. In *Proceedings of the Eleventh ACM Workshop on Challenged Networks (CHANTS'16)*. ACM.

[13] Zongheng Ma, Saeed Mirzamohammadi, and Ardalan Amiri Sani. 2017. Understanding sensor notifications on mobile devices. In *Proceedings of the 18th International Workshop on Mobile Computing Systems and Applications (HotMobile '17)*. ACM.

[14] Saeed Mirzamohammadi and Ardalan Amiri Sani. 2018. Viola: trustworthy sensor notifications for enhanced privacy on mobile systems. *IEEE Transactions on Mobile Computing* (2018).

[15] Unkyu Park and John Heidemann. 2011. Data muling with mobile phones for sensornets. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems (SenSys'11)*. ACM.

[16] Rijurekha Sen, Abhinav Maurya, Bhaskaran Raman, Rupesh Mehta, Ramakrishnan Kalyanaraman, Nagamanoj Vankadhara, Swaroop Roy, and Prashima Sharma. 2012. Kyun queue: a sensor network system to monitor road traffic queues. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems (SenSys'12)*. ACM.

[17] Gilman Tolle, Joseph Polastre, Robert Szewczyk, David Culler, Neil Turner, Kevin Tu, Stephen Burgess, Todd Dawson, Phil Buonadonna, David Gay, and others. 2005. A macroscope in the redwoods. In *Proceedings of the 3rd international conference on Embedded networked sensor systems (SenSys'05)*. ACM.

[18] 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 (HotMobile'15)*. ACM.