DisruptaBLE: Opportunistic BLE Networking

Patrick Rathje¹, Olaf Landsiedel¹,²
¹Kiel University, Germany
²Chalmers University of Technology, Sweden
{pra, ol}@informatik.uni-kiel.de

Abstract—Bluetooth Low Energy (BLE) is the prevalent IoT radio technology and perfectly suited for mobile and battery-driven applications. However, it is not designed for intermittent connectivity and opportunistic networking. Hence, the vast infrastructure that BLE-equipped devices such as smartphones, wearables, and sensors provide, remains untapped, even with its potential for data collection, sharing, or emergency communication in disaster scenarios. This paper introduces DISRUPTABLE to unfold this potential: A universal BLE-based store-and-forward architecture for delay-tolerant and opportunistic networking. Tailored to the resource constraints of IoT nodes and the feature set of BLE, DISRUPTABLE enables opportunistic interactions between BLE-equipped devices, providing a resilient network even when established communication over cellular networks or Wi-Fi fails. In our evaluation, we show that in a highly dynamic pedestrian scenario in downtown Stockholm, broadcasts reliably inform pedestrians in 7.1 seconds, while unicast messages arrive within 20 minutes in 48.1% of cases.

Index Terms—DTN, Opportunistic Networking, Bluetooth Low Energy, Delay-Tolerant Internet of Things, Internet of Things

I. INTRODUCTION

In 2021 alone, more than three billion devices equipped with Bluetooth Low Energy (BLE) were shipped, making BLE an omnipresent radio technology for low-power IoT devices [1]. BLE devices, such as smartphones, wearables and sensors, reside in households, offices, and even in our pockets, accompanying us throughout our daily lives. Thus, BLE-enabled devices offer an infrastructure for citywide networks enabling applications in delay and disruption tolerant networking (DTN) and Opportunistic Networking [2] like the collection of sensor data [3] or emergency communication in disaster scenarios when established communication over cellular networks or Wi-Fi fails [4].

Reflecting on recent events such as the flooding in Germany during summer 2021 or the recent Tsunami in Tonga at the beginning of 2022 underlines the need for emergency communication. During these catastrophic events, cellular and Wi-Fi networks failed in the affected regions, and it took days to weeks until communication was restored to a basic level.

We suggest BLE as a perfect candidate for resilient DTN communication, especially suited to disaster scenarios where wireless access networks and often also power fail: BLE sensors can operate for weeks on battery, and even smartphones (just using BLE) can operate for days without charging. However, BLE merely provides point-to-point communication and mesh networking between known devices. Opportunistic interactions between unseen devices which are common in mobile DTN scenarios are yet to be utilized. Moreover, BLE lacks a DTN architecture tailored towards its communication primitives and resource constraints. Further, evaluation in realistic environments constitutes a challenge in DTN research and remains one for BLE, as established evaluation environments for low-power wireless networking such as static testbeds fail to reproduce the dynamic topologies created by the mobility in DTN environments.

This paper closes these gaps and introduces DISRUPTABLE: disruption-tolerant networking tailored to the specifics of resource-constrained platforms and the communication primitives BLE provides. For this, it matches and augments BLE’s feature set to DTN semantics and, in particular, focuses on neighbor detection, selection, and wireless data communication in resource-constrained environments. Targeting challenging, mobile DTN scenarios, that we cannot replicate in a static testbed, we introduce a system-level simulation architecture. Running the same code-base and protocol stack in simulations and actual hardware, we calibrate a reproducible simulation environment that inherently captures the details of BLE communication. Our evaluation utilizes both large-scale simulations and device-to-device interactions in a static scenario, executed both on real and simulated hardware. Within the calibrated BLE radio medium, we further analyze the forwarding behavior in mobility simulations featuring traces from pedestrians in downtown Stockholm.

Overall, this work contributes the following:

1) We introduce DISRUPTABLE, which tailors delay-tolerant networking to the specifics of resource-constrained platforms and protocols, enabling DTN for billions of IoT devices.¹
2) We provide a system-level simulation architecture for accurate, mobility-based experiments, allowing DISRUPTABLE to run the same code-base in simulations and deployments on actual hardware.
3) We evaluate DISRUPTABLE in a static testbed and via mobility simulations. For one, we simulate broadcasts and unicasts in a calibrated BLE radio environment, featuring mobility from the KTH walkers dataset. In this highly dynamic scenario with pedestrians joining and leaving in downtown Stockholm, a broadcast reliably reaches all devices in 7.1 seconds on average.

¹Source code available at https://github.com/prathje/DisruptaBLE

This work is supported by the Federal Ministry of Health based on a resolution by the German Bundestag.
II. RELATED WORK

Bluetooth itself is not new in the DTN domain as the Haggle project envisioned communication between devices in one’s pocket using technologies such as Wi-Fi or Bluetooth [5]. While it used to support bundle transport over RFCOMM channels on Bluetooth Classic, the DTN reference implementation did neither support dynamic connectivity nor BLE and completely removed support in the newer version [6]. Solutions for mesh networking, available for Wi-Fi as IEEE 802.11s and for BLE [7], are not designed for intermittent connectivity. Thus, the IoT domain contains multiple examples for custom, application-specific DTN implementations, ranging from emergency communication [4] over the collection of sensor data [3] to opportunistic device-to-device interactions [8]. Still, no general DTN or Bundle Protocol approach exists for BLE while there are solutions available for IEEE 802.15.4 [9] or CoAP [10]. Building on established DTN technology like μD3TN, DISRUPTABLE integrates into existing networks [11], expanding opportunistic connectivity to BLE.

Higher-level simulators provide an evaluation ground for DTN and Opportunistic Networks [12]. With a large variety of available movement patterns covering generated as well as real-life datasets, simulators like the ONE [13] focus on the mobility aspect while simplify transmissions and run code written explicitly for simulations. LEPTON, an emulation platform for opportunistic networking, fills the software gap and builds on the emulation of real-code to support the development of opportunistic networking middleware and applications [14]. In the same mentality, our simulations incorporate device-ready code using the virtualization of resource-constrained devices. Moreover, we execute the full BLE-stack in a calibrated BLE radio environment.

III. BACKGROUND

A. Opportunistic Networks

Characterized by temporary, unscheduled connections, Opportunistic Networks are a special kind of Delay/Disruption Tolerant Networks (DTN) in which stable end-to-end links are generally not available [15]. Covering applications like sensor data collection or emergency communication, participating devices (or nodes) temporarily store and forward data until the data eventually reaches its destination, as seen in Figure 1. Forwarding is thus based on a mixture of dynamic topology changes due to mobility and the data stored on individual nodes. Two opposing extremes of this forwarding design space are Epidemic Routing and Direct Transmission. Epidemic Routing floods the entire network by sending a copy to each available neighbor, requiring extensive resources, as devices continually forward copies to all neighbors to minimize the delivery delay [16]. In contrast, Direct Transmission only transmits on direct contact to the destination and hence minimizes resource costs but induces potentially high delays, i.e., the time until the sender device eventually encounters the target itself [17]. As a trade-off between low delay and low memory-overhead, the Spray-and-Wait algorithm introduces an initial, limited number of copies, which are then forwarded using Direct Transmission [18]. Naturally, the mobility of nodes and metrics like group membership or recent contact times are key factors in the forwarding process and motivated the design of numerous routing algorithms [15]. Most of the algorithms are designed for and evaluated in abstract simulation environments [14]. As DISRUPTABLE targets practical systems, we focus on the basic mechanisms and employ Epidemic Routing for broadcast and Spray-and-Wait for unicast forwarding.

B. Bundle Protocol

Motivated by the requirements of interplanetary networks [19], the Bundle Protocol [20] specifies a DTN architecture and data serialization format to store-and-forward data under challenging network conditions. Nodes form an overlay network that capsules and exchanges data in “bundles” irrespective of underlying network technologies. In this overlay network, Endpoint Identifiers (EID) address nodes across network boundaries. Convergence Layer Adapters (CLAs) individually handle lower-level network technologies such as TCP [21], SCTP [22] or IEEE 802.15.4 [9].

C. Bluetooth Low Energy

Bluetooth Low Energy (BLE) is a short-range communication protocol with a particular focus on energy efficiency. With data rates up to 2 Mbit/s, many smartphones, wearables like fitness trackers, and other IoT sensor devices rely on BLE for data exchange [23]. BLE includes neighbor discovery based on advertisement beacons, where each device periodically advertises its services allowing nearby devices to listen for connection candidates. At its core, the Logical Link and Adaption Protocol (L2CAP) resides on top of BLE’s Link-Layer and handles multiplexing, segmentation, and reassembly for the upper layers. With its Credit-Based Flow Control mode, BLE allows applications to use L2CAP connection-oriented channels directly, allowing for bidirectional data streams. While being designed for energy efficiency, BLE’s overall energy consumption depends on factors like the advertisement interval and amount of data. For one, a BLE sensor node can transmit around 10 Mbit per day for a year running on a coin cell [3], which is sufficient for battery-driven DTN networking in, e.g., the Internet of Things or emergency communication.
IV. DISRUPTABLE: BLE-BASED DTN

The design of DISRUPTABLE needs to bridge the gap between the semantics of DTN, i.e., the abstract notion of sending and receiving in a delay-tolerant manner, and the BLE communication principles of connections and advertisements. In addition, from a practical perspective, it should ensure easy integration into existing DTN solutions and seamless execution within the BLE-ecosystem, even on resource-constrained devices. Figure 2 illustrates our architecture: we use standard BLE features for neighbor discovery, neighbor selection, and bundle transport and integrate them into the Bundle Protocol (BP), ensuring DTN compatibility. Regarding power limitations, we augment and optimize the neighbor selection. By building upon microcontroller-ready frameworks, we tackle resource constraints from the ground up.

While the BP streamlines internal bundle processing and applications interfacing, it employs CLAs for specific network technologies to handle neighbor discovery and the bidirectional transport of serialized bundles. We, therefore, focus on the design of a CLA for BLE. We first cover neighbor discovery and selection, followed by bundle transport. We continue with forwarding mechanisms and the system simulation architecture. An outline of our implementation completes this section.

A. Neighbor Discovery and Selection

In the dynamic environments of opportunistic networks, finding promising connection candidates requires continuous neighbor detection. But a potential connection does not guarantee to be stable – as devices might move apart – or valuable – as the information to share is already known. As neighbor detection and even more maintaining connections are power-hungry tasks for any battery-driven IoT device, devices should only establish connections if it seems beneficial: they need to filter out unfruitful candidates.

For neighbor discovery, we employ BLE’s existing advertisement mechanism and extend the advertisement beacons: In DISRUPTABLE, devices simultaneously scan and advertise themselves using a specific service identifier, acting simultaneously as a BLE client and peripheral, stretching beyond BLE’s typical use-case. For this, we extend advertising beacons with the device’s overlay network address, its EID, allowing identification within the overlay network. This allows devices to detect desired neighbors without the need to connect to every device. We reuse the MAC address, already included in the advertisement data, as the CLA address, serving as a transport address for the Bundle Protocol.

In addition to the fundamental discovery mechanism, we filter out connection candidates with potentially uninteresting data based on the advertising data: Each device hashes its offered bundle information and concatenates the hashes using bitwise XOR to a single summary hash which we denote as Offer Hash. This Offer Hash resides in the advertisement data and thus characterizes the offered bundles independent of their order and number. Each device keeps a history of hashes, allowing it to derive whether the advertising device has potentially new, hence interesting bundles to share with them. Devices extend their history with their own advertised Offer Hash as well as the hashes gained through successful connections to other devices. Saved in a circular buffer, new Offer Hashes overwrite the oldest entries. As opposed to ignoring recent devices after connections, this content-driven approach requires no timeout and enables the categorization of previously unseen devices as they might share already known data. As Figure 3 illustrates, due to flags and service identifiers, out of the 31 available Bytes in BLE’s standard advertisement beacons, just 24 Bytes remain customizable. DISRUPTABLE uses 8 Bytes for the Offer Hash and leverages the remaining 16 Bytes for the overlay address or, e.g., its hash representation.

Devices do not initiate a connection to a neighbor with a known Offer Hash, but they still accept incoming connections as the other device could need data from them, resulting in a productive pull [24]. However, when devices try to connect to pull new information from other nodes, they act on the assumption that some bundles should be forwarded to them. While this assumption suits scenarios well in which the forwarding is unspecific, like Epidemic Routing, with very specific forwarding such as Direct Transmission, most devices would vainly attempt to pull information, wasting resources on both ends. A variant with a productive push, for one based on the EID, could improve this scenario. Still, even in this suboptimal scenario, the Offer Hash prevents connection to recent devices until either the other device offers new bundles or the respective Offer Hash gets evicted from the history.

B. Bundle Transport

While the Bundle Protocol oversees storing, serializing, and parsing bundles, our BLE Convergence Layer Adapter accounts for the actual transport, including connection setup. Devices first connect to valuable, i.e., non-filtered neigh-
bors and finish the setup process by establishing an L2CAP connection-oriented channel. Devices set their advertisements as non-connectable whenever they cannot handle more incoming connections. The L2CAP channel between the two devices allows for bidirectional data transfer comparable to TCP. We let each side send the length before every serialized bundle, allowing the receiver to split the data stream back into individual bundles [21].

C. Bundle Forwarding

Forwarding algorithms employ a strategy to identify the bundles to forward to a certain neighbor. We define a router component that provides Epidemic Routing as well as Spray-and-Wait forwarding policies. The design and implementation of DISRUPTABLE are, however, generic and directly support other policies as a drop-in replacement. Inspired by Epidemic Routing [16], we let devices offer stored bundles to each other using summary vectors: devices hash each bundle, allowing space-efficient identification of bundles as a list of hashes. After establishing a connection, two nodes first transmit the respective summary vector as an offer, as seen in Figure 4. The other device filters out the offer and requests only wanted bundles by sending a summary vector as a request. We hence incorporate knowledge about already present bundles and prevent the duplicate transport of possibly large bundles. Afterward, each device transmits the requested bundles over the BLE adapter.

D. System Simulation

Opportunistic networking scenarios incorporate a highly dynamic network topology. As these dynamics stem from the inherent mobility, we cannot replicate this environment in a static testbed. Instead, we devise a system simulation architecture that supports mobile nodes yet captures important low-level characteristics of DISRUPTABLE. Using a hardware abstraction layer, we execute the same device-ready code for virtual nodes, executing the complete BLE protocol stack within a virtual 2.4 GHz PHY layer. This setup captures both implementation details of DISRUPTABLE and low-level BLE characteristics, including BLE advertisements, neighbor discovery, and connection setup. In addition to the actual transmissions, our incorporated PHY layer simulates the respective interference as well as background noise and Wi-Fi interference. Due to DISRUPTABLE’s focus on resource-constrained devices, each device in the simulation only incorporates a small memory footprint, allowing simulations with hundreds of devices. For mobility, our toolchain offers a generic interface with granular control over device positions, supporting traces such as the KTH Walkers dataset [25].

E. Implementation

The Zephyr RTOS constitutes the foundation for our implementation, a real-time operating system for connected resource-constrained devices that features an open-source, certified Bluetooth stack [26]. Its large number of supported chips and platforms makes it the ideal cornerstone for the implementation. For compliance with other DTN solutions, we integrate and extend µD3TN, an interoperable Bundle Protocol implementation, allowing DISRUPTABLE to handle generic data bundles from arbitrary sources. We limit the bundle exchange to just a single active connection at a time and set the timeout to four seconds.

Our system simulation builds on BabbleSim, a modular physical layer simulator [27]. BabbleSim offers a deterministic and detailed BLE radio environment that has been used for BLE advertising optimization [28].

V. EVALUATION

For the evaluation of DISRUPTABLE, we first introduce the setup, our testbed, and parameters of our simulated radio environment. We continue by evaluating node interactions in the static testbed and its virtual, simulated copy. Next, we analyze the behavior in a highly mobile scenario based on traces from the KTH Walkers dataset [25]. We close our evaluation with a discussion of the results.

A. Setup: Testbed and System Simulations

We build on the nRF52840 as our target platform, a low-power SoC from the widely deployed nRF52 family that features BLE 5.2, an ARM Cortex-M4 64 MHz processor, 1 MB Flash, and 256 KB RAM. Our testbed consists of 20 static nRF52840 devices, spread over multiple rooms on a single floor in an office building, depicted in Figure 5a. Our simulations are based on BabbleSim and a Zephyr-based hardware abstraction layer for the nRF52 platform. We minimize the abstraction level as we run the same code, including the complete BLE protocol stack, on both the real hardware and in simulations. For the simulated radio environment, we set the background noise level to $-96.4\, \text{dBm}$ [29]. In addition, we add Wi-Fi interference at $-70\, \text{dBm}$ (25% duty cycle) and set the path loss exponent to 3.6. Table I summarizes the parameters. We release the respective source code and the simulation toolchain providing a reproducible evaluation ground for DTN experimentation to the public.

B. Lower-Level Interactions and Simulation Calibration

We use the testbed for low-level evaluations of device-to-device interactions in DISRUPTABLE, including performance metrics for neighbor discovery, connection setup, and data exchange. Furthermore, we compare the testbed against the simulated radio environment, revealing the introduced layer of abstraction. An interaction starts with the reception of an
advertisement, followed by the setup time of both the BLE connection and the L2CAP channel. After the setup completes, devices exchange bundles, the first four being the summary vector exchange, followed by potential payload bundles.

For this experiment, we place 20 virtual nodes statically to the respective device positions in the testbed (see Figure 5a). The simulations and the testbed execute the same scenario: at startup, each node generates a unicast bundle of 1024 Bytes in size and forwards a replica via Spray-and-Wait to each of its available neighbors. We run and simulate for 30 minutes with overall 10 repetitions. We collect the received bytes in size and forwards a replica via Spray-and-Wait to at startup, each node generates a unicast bundle of 1024 Bytes. We run no less than five iterations per arrival rate for unicasts. We set the nominal payload size of bundles to 1 kB, with contact durations of more than 256 seconds occurring in 25% of cases. We place one designated central node statically at the area’s center. This node acts as the origin of broadcasts to the network and as the sink for unicasts issued by all other nodes. In the broadcast scenario, we distribute data from the central device to the whole network, i.e., all other devices. In the unicast case, each device addresses a message to the central node. For forwarding, we use Epidemic Routing for broadcasts and Spray-And-Wait with 20 additional replicas for unicasts. We set the nominal payload size of bundles to 1024 Bytes. We run no less than five iterations per arrival rate and scenario for up to 30 minutes, overall simulating at least 141 (Low), 412 (Medium) respectively 756 (High) virtual devices per run.

We measure the reception rate of broadcast bundles and determine the overall rate of active devices in the area that received this bundle. Helgason and Jónsson ran experiments in the same area, assuming that if devices stay within 20

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td></td>
</tr>
<tr>
<td>Bundle Size</td>
<td>1 kB</td>
</tr>
<tr>
<td>Offer Hash History</td>
<td>128</td>
</tr>
<tr>
<td>BLE Config.</td>
<td></td>
</tr>
<tr>
<td>Mode</td>
<td>LE 2M</td>
</tr>
<tr>
<td>Connection Interval</td>
<td>7.5 ms</td>
</tr>
<tr>
<td>TX MTU</td>
<td>245 B</td>
</tr>
<tr>
<td>TX Power</td>
<td>0 dBm</td>
</tr>
<tr>
<td>Radio Env.</td>
<td></td>
</tr>
<tr>
<td>Background Noise</td>
<td>−96.4 dBm</td>
</tr>
<tr>
<td>Wi-Fi Interference</td>
<td>−70.0 dBm</td>
</tr>
<tr>
<td>Wi-Fi Duty Cycle</td>
<td>25%</td>
</tr>
<tr>
<td>Path Loss Exponent</td>
<td>3.6</td>
</tr>
<tr>
<td>Mobility Scenario</td>
<td></td>
</tr>
<tr>
<td>Mobility Trace</td>
<td>KTH Walkers</td>
</tr>
<tr>
<td>Mean Velocity</td>
<td>1.3 s⁻¹</td>
</tr>
<tr>
<td>Mean Arrival Rates</td>
<td></td>
</tr>
<tr>
<td>[nodes/min]</td>
<td>8.4 (Low)</td>
</tr>
<tr>
<td></td>
<td>25.2 (Medium)</td>
</tr>
<tr>
<td></td>
<td>42.0 (High)</td>
</tr>
</tbody>
</table>

TABLE I: Simulation Parameters

Fig. 4: Bundle Exchange between two BLE devices: Device A receives an advertisement from device B and initializes a connection. As soon as the connection and the channel are established, exchanged summary vectors offer respectively request a set of bundles from the other party. Subsequently, requested bundles are transmitted. The connection finally times out.

Overall, the simulations perform similarly (or worse) than the actual testbed. As the simulations do not incorporate unique obstacles like walls, the results show less deviation. Thus, the simulation results primarily depend on the distance between nodes. We accept the deviations to limit simulation complexity.

C. Opportunistic BLE Networking in Downtown Stockholm

For a challenging DTN scenario, we dissect the forwarding process using the KTH walkers dataset within our calibrated radio environment [25]. Covering an area of approximately 5872 m² (see Figure 6a), this dataset provides accurate positional information of hundreds of walkers in Östermalm, downtown Stockholm, resulting in a challenging opportunistic DTN setting. Based on the traces, we simulate walkers with nRF-devices, running DISRUPTABLE in a grid of streets and analyze the forwarding delay for both a broadcast and a unicast scenario. In this high mobility setting, nodes arrive according to a Poisson process, walk with a mean speed of 1.3 m/s, and enter respectively leave the area through one of fourteen passages. We vary the arrival rate to achieve overall mean arrival rates of 8.4 (Low), 25.2 (Medium) to 42.0 (High) nodes per minute. Figure 6b displays the development of active devices for each arrival rate. Illustrated by Figure 6c, the inter-contact time probability, i.e., the probability that two devices stay within a range of 20 m for a certain time, stays similar across arrival rates: short-term contacts are the most probable with contact durations of more than 20 seconds occurring in less than 30% of cases. We place one designated central node statically at the area’s center. This node acts as the origin of broadcasts to the network and as the sink for unicasts issued by all other nodes. In the broadcast scenario, we distribute data from the central device to the whole network, i.e., all other devices. In the unicast case, each device addresses a message to the central node. For forwarding, we use Epidemic Routing for broadcasts and Spray-And-Wait with 20 additional replicas for unicasts. We set the nominal payload size of bundles to 1024 Bytes. We run no less than five iterations per arrival rate and scenario for up to 30 minutes, overall simulating at least 141 (Low), 412 (Medium) respectively 756 (High) virtual devices per run.

We measure the reception rate of broadcast bundles and determine the overall rate of active devices in the area that received this bundle. Helgason and Jónsson ran experiments in the same area, assuming that if devices stay within 20
For Unicasts, we gather the times from the arrival of a node until its message reaches the central node, starting from nodes that arrive after 3 minutes. Because we deploy the Spray-and-Wait algorithm, each node forwards up to 20 copies of its message, increasing the chance that one of the replicas eventually reaches the central node. Figure 9 illustrates the average number of replicas and the average rate of delivered unicast bundles. With each increase in the number of nodes, devices were able to share more replicas. As time goes by, nodes carrying replicas leave the area, reducing the mean number of copies in the region after 20 minutes to 1.6 (Low), 2.4 (Medium), and 0.7 (High). Even though the likelihood to reach the central node increases with more replicas, results show that this does neither guarantee a faster nor a reliable delivery in our scenario: With a mean delivery rate of 48.1% meters for at least 20 seconds they exchange the broadcast information [31]. For comparison, we use their parameters and calculate reference values for each trace. Displayed by Figure 7, the reception rate slowly increases as more and more nodes arrive in the area. After the initial ten minutes, the different arrival rates resulted in overall average reception rates of 85.7% (Low), 96.8% (Medium), 98.2% (High) for the rest of the simulations. With an increasing number of devices that support the forwarding, the Medium and High scenarios achieve stable reception rates while the rates in the Low scenario fluctuate. Compared to the reference bounds, all scenarios achieved reception rates above the reference, despite being limited to just a single active connection. Of the nodes that arrived after the first ten minutes, all nodes received the broadcast after 7.1 seconds on average during their stay in the High scenario. In the Medium and Low scenario, on average, 99.5% (Medium) and 92.7% (Low) obtained the message, after 11.7 seconds (Medium) respectively 37.3 seconds (Low). In our simulations, the complete forwarding process of the broadcast message between two devices, i.e., the time from reception of the advertisement to the reception of the payload bundle, took 2.3 seconds on average, requiring a fraction of the assumed 20 seconds reference value [31] (see Figure 8). Yet, as Figure 8 depicts, the distance usually stayed below 20 meters, matching our reference [31].
after 20 minutes, the Low arrival rate featured the highest rate, followed by 47.9% (Medium) and 41.5% (High).

To prevent unproductive, energy-consuming connections altogether, devices summarize their bundles as a hash in the advertisements. Each device stores a history of hashes and only connects upon receiving unseen values, trying to fetch new data. We gather connection statistics from the Low and High arrival rates in both forwarding scenarios and distinguish five different outcomes: (1) the connection failed, (2) the summary vector exchange did not finish, (3) an unproductive connection (no payload bundles to be exchanged), (4) the connection was interrupted while transmitting payload bundles, and (5) a successfully finished exchange. As seen in Figure 10, on average, the broadcast scenarios involved only a fraction of connections, having about 150 in the Low and roughly 1450 in the High setting, opposed to approximately 1050 (Low) and around 22000 (High) for unicasts. No unproductive connections occurred in the broadcast case, showing the efficacy of the incorporated Offer Hash. As the unicast scenario contains concurrent bundles that are selectively forwarded, devices pull other nodes more frequently but can still ignore devices after the summary vector exchange. The rate of failed connections increases with higher arrival rates and shows potential for further optimizations.

We further analyzed times spent in power-hungry connections (omitting the central node). Depicted by Figure 11, our results show that unicast forwarding significantly increases the overall time for devices from 9.0 seconds (Broadcast Low) to 46.0 seconds (Unicast Low) respectively from 13.5 seconds (Broadcast High) to 115.9 seconds (Unicast High).

D. Evaluation Discussion

Overall, our results underline that DISRUPTABLE is an efficient architecture and implementation for DTN and opportunistic networking via constrained BLE devices. The broadcast scenario shows efficient propagation, achieving higher reception rates than the reference values. Yet, unicast forwarding leaves room for improvements, e.g., by using a productive push, based on the advertised EID, and thus a tighter integration of neighbor selection and forwarding.

We did not model where the pedestrians carry BLE-devices
As these effects are best captured using real-life experiments. Though their impact is potentially negligible in large-scale simulations [32].

Despite these limitations, DISRUPTABLE’s simulation architecture offers a calibrated and reproducible evaluation ground and incorporates low-level BLE characteristics that are not found in higher-level DTN simulators like the ONE simulator [13] or LEPTON [14].

VI. Conclusion

With DISRUPTABLE, we lay the foundation for disruption-tolerant and opportunistic networking over BLE, enabling application scenarios like sensor data collection or disaster networks. DISRUPTABLE bridges the gap between BLE’s feature set and the Bundle Protocol, provides broadcast and unicast forwarding, and introduces optimizations targeting neighbor selection. We analyze device-to-device interactions and the overall forwarding process in BLE-specific radio environments using a testbed deployment and calibrated mobility simulations of walkers in downtown Stockholm. Within our experiments of different arrival rates, an epidemic broadcast message reaches 92.7% to 100% of devices with an average delay of 36.3 respectively 7.1 seconds. Unicast messages show delivery rates between 41.5% and 48.1% within 20 minutes.

Future directions include field testing, a detailed evaluation of the energy efficiency and an investigation of the incentives for opportunistic transport.

REFERENCES