Heuristic Scheduler for Proximity Data Exchange

This paper was downloaded from TechRxiv (<https://www.techrxiv.org>).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

04-06-2021 / 02-08-2021

CITATION

T S Narayanan, Hari (2021): Heuristic Scheduler for Proximity Data Exchange. TechRxiv. Preprint.
<https://doi.org/10.36227/techrxiv.14730648.v2>

DOI

[10.36227/techrxiv.14730648.v2](https://doi.org/10.36227/techrxiv.14730648.v2)

Heuristic Scheduler for Proximity Data Exchange

Hari T.S. Narayanan, Senior Consultant
NetTools Consulting, ts.hari@gmail.com,
Mobile: +91 9841289525

Abstract – A digital contact tracing system collects, stores, and computes proximity distance and duration to identify the contacts for contagious diseases like SARS COVID-19. Most of the currently deployed contact tracing solutions are built with smartphones. The Bluetooth Low Energy (BLE) in smartphones is used for exchanging proximity data. This exchange method can be either intrusive or non-intrusive. In an intrusive method, the data exchange takes place after establishing a BLE connection to another smartphone. In a non-intrusive method, proximity data is scanned from the periodic BLE broadcast messages from a smartphone. The BLEs operate in Observer and Broadcaster roles in this exchange. Irrespective of the method, there can be collisions while accessing the media. Collision could reduce the reliability of data exchange thus reducing the validity of the data collected. In this paper we present a heuristic for the broadcast and scanning schedules of non-intrusive methods of proximity data exchange. The objective of this heuristic is to optimize scanning reliability and to conserve power. A model is built to quantify reliability using a generalization of the Birthday Problem. The schedule adapts to changing loads with optimal agility. The heuristic is designed to be simple to conserve smartphone resources. It can coexist with other BLE services.

Index Terms - Contact Tracing Apps, Smartphone, Bluetooth Low Energy, Proximity Data, Heuristic Scheduling, Reliability, Birthday Problem, SARS COVID-19.

1. INTRODUCTION

A contact tracing system collects, stores, and computes data to identify the *contacts* and the *cases* for a contagious disease. A subject is a *case* when the subject is having an infection specific symptoms or signs. A subject is a *contact* if the subject has likely contracted the virus but asymptomatic. The healthcare authority in a country develops the necessary guidelines which define the terms *contact* and *case* for an infection. The guidelines developed for an infection by a country can differ from others in subtle ways. In this paper, the guidelines released by the Centers for Disease Control (CDC) and Prevention, US [1] are assumed for illustrations. The CDC guidelines suggest that if a subject spends more than 15 minutes within 6 feet of SARS-2 COVID-19 infected person, then the subject is a *contact* and a potential *case*. A *contact* can turn into a *case* within 14 days of becoming a contact and not likely after this period. The complete guidelines can be accessed from CDC's official site [1].

A comprehensive tutorial and survey of digital contact tracing can be found in [2]. All currently deployed contact tracing solutions [3-7] are built with smartphone proximity service. The Bluetooth Low Energy (BLE) [8] adapter in smartphones is used in exchanging proximity data. This adapter can be configured with a schedule to advertise its

presence and to scan for advertisements from other smartphones within its operating range. The BLE advertisement provides the presence data of an advertising smartphone. The distance between two BLE devices is computed as a function of differential signal strength between BLE transmitter and receiver. The duration is computed by consolidating and correlating the periodically collected proximity data. There are solutions where this data is complimented by tracking the subject's location using wearable, cellular or GPS (Global Positioning System) technologies. BLE devices are built for peer interactions unlike other wireless technologies like WiFi and Cellular. A BLE in a smartphone can be configured to advertise and scan periodically besides occasionally connecting to other BLE devices for rendering some services. The only modification required for the periodic exchange of proximity data is the definition of Universally Unique ID (UUID) [9] to differentiate proximity payload from other payload types. There is an UUID already suggested for this purpose in one of the existing proposals[5-7]. The scheduler described in this paper can be a key component of the proximity service hosted on smartphones. This service can be used by any App, and it can be hosted with any BLE device. The illustrations in this paper, however, are developed for contact tracing App.

There are several contact tracing solutions [10], some are for closed user groups, some are for the

residents of a country [3, 4], and some are open solutions with no clear boundary defined [5-7]. There are two design approaches to processing proximity data. In one approach, the data is uploaded and processed in a centralized server [3,4] to identify *contacts*. In the second one the data is processed in every registered smartphone, thus distributing the processing load [5-7]. Both have their pros and cons [2, 11]. The design presented in [12] offers flexible distribution of processing load and includes a federated solution for a seamless interworking of several distributed or centralized systems.

In all contact tracing applications, the user is identified by *proximity identifier (PID)*. The operational PID is rotated, and thus referred to as the *Rotating Proximity Identifier (RPID)*. The desirable property of PID is presented in [11]. One such property is anonymity (of the user). We classify the anonymity supported by contact tracing solutions into three categories: *no-risk* anonymity, *low-risk* anonymity, and the trivial category of *no-anonymity*. There are applications where no-anonymity is acceptable. In no-risk anonymity, the subject's identity is a random value that identifies the smartphone user anonymously [4-7]. There is no correlation between the identifier and its subject. This random identifier reveals the subject's identity only under controlled conditions. In low-risk anonymity, the identifier is an encrypted value of the subject's identity[11]. The robustness of this anonymity can be quantified using that of its ciphering. A strong ciphering could offer the necessary level of anonymity.

There are two methods for exchanging proximity data; it is done either *intrusively* or *non-intrusively*. In the intrusive methods [4], a smartphone opens a BLE connection to another smartphone after scanning and finding it; and gets the proximity data exchanged within the connection. In the *non-intrusive* methods[5-7], the data is scanned and extracted from the periodic advertisement broadcasts sent by BLEs. The individual connection to every other BLE and among every pair of BLE requires more power and time. This limits the scalability and reliability of intrusive exchange. The *non-intrusive* method on

the other hand scales better with broadcast-scanning combination. The absence of connection also makes this exchange more secure. However, it is limited by its 31-byte payload. The next significant version of Bluetooth (5.0) supports a much larger payload; however, its user base is insignificant at this point of time. Irrespective of the exchange method used, the advertisements from two or more smartphones could collide. This needs to be addressed by these methods.

A BLE device can be configured to operate in one or a certain permitted combinations of the following four roles:

1. **Broadcaster:** An advertiser with no intention to host a connection request.
2. **Observer:** A scanner with no intention to initiate a connection request.
3. **Peripheral:** An advertiser with an intention to connect. Operates as a slave to a device with a Central role.
4. **Central:** A scanner with the intention to connect. It can operate as a master to a single or multiple Peripheral

A Peripheral device advertises only when it needs to be discovered and connected. It stops advertising once connected to a Central device. The Central device may also stop scanning once it connects to the Peripheral device. On the other hand, a Broadcaster and an Advertiser may perform their task perpetually and periodically. A BLE device can advertise and scan while hosting multiple BLE connections. When multiple BLEs operate within mutual operating range, their broadcasts can collide. As a result of this, the time to successfully scan to locate a specific BLE and the latency to establish the connection could become non-deterministic. There are popular BLE specific analytical and experimental models [13-15] to find the expected latency for a pair of BLEs operating in Central and Peripheral roles respectively. This latency includes the time taken by the Central device to scan, discover, and connect to a specific Peripheral. This is modelled using the probability of success for a sequence of bursty events that culminates in a connection within a certain duration.

The objective and scope of the model presented in this paper is different from the above models. The scanning reliability of a BLE Observer embedded to a smartphone under the following conditions is modelled. There is a dynamic set of BLEs within the operating range of an Observer. The Observer and the BLEs in the set operate predominantly in peer mode. That is, each BLE advertises its presence periodically and does periodic scanning of peer advertisements. The advertisement can be from any of the peer devices and successful scanning is the event. There is no subsequent connection attempt. A model is built to measure the scanning reliability of an Observer under this operational scenario. A heuristic scanning scheduler is built with the model to improve the reliability under varying load conditions. The scheduler adapts to dynamically changing load conditions and attempts to optimize the power consumption. It can operate under the constraints imposed by the BLE adapter, BLE library, and within the resource restrictions of smartphones. This scheduler can offer this as a service to all proximity and location applications hosted on smartphones. It is assumed that this is the only BLE service that operates in Observer and Broadcaster roles. All other BLE operations are restricted to Central and Peripheral roles. The scheduler can be used with any design of contact tracing that employs non-intrusive exchange.

The rest of the paper is organized into five sections. Section 2 describes the Bluetooth LE discovery features by highlighting the timing and other constraints. Section 3 describes the content of the proximity message exchanged using our heuristic. Section 4 and Section 5 are the core sections of this paper. In Section 4, a reliability model for scanning is developed. This model is used to develop the heuristic for scheduling broadcast and scanning in Section 5. Section 5 also addresses other related aspects of the scheduler. Section 6 concludes the paper.

2. BLE NEIGHBOUR DISCOVERY PROCEDURE

Bluetooth technology is built to support neighbour discovery. The Neighbour Discovery Procedure (NDP) of Bluetooth includes two features: Advertising and Scanning. The scheduling

presented in this paper is developed for Bluetooth LE (BLE), the version of Bluetooth with the largest user base at present. Evolving the scheduler for Bluetooth 5.0, the current version of Bluetooth, is presented at the end of Section 5. The Bluetooth 5.0 is backward compatible with BLE.

The Bluetooth Low Energy (BLE) shares the 2.4 GHz ISM band with WiFi and other wireless technologies. This band is divided into 40 channels on 2 MHz spacing from 2.4000 GHz to 2.4835 GHz by BLE, starting at 2402 MHz as shown in Figure 1. Data rate of 1 Mbps is supported on each channel. The three dispersed channels 37, 38, and 39 are used only for sending advertisement frames; the rest of the channels 0-36 are used for data. The scope of the scheduler is restricted to the events of the advertisement channels.

Fig. 1. BLE Advertisement and Data Channels

When advertising, a BLE Peripheral device transmits the same message on the 3 advertising channels, one after the other as shown in Figure 2. An Observer or Central device that is looking for a Peripheral will scan those channels for advertisements. An advertising message can be *directed* or *undirected* broadcast. The directed advertisement targets a single receiver; the undirected one targets no specific receiver.

Fig. 2. BLE Advertisement[13]

There is more than one type of advertisement message. The one that is of interest here is an undirected advertisement `ADV_IND`. The `ADV_IND` frame format is described in Section 3.

Each advertisement event includes sending an *advertisement* frame on all the three channels (37, 38, and 39) in order. The interval (*AdvInterval*) between two successive advertisement *events* initiated by a Broadcaster can be from 20 milliseconds (ms) to 10.24 seconds, in steps of 0.625 ms. If two or more Broadcasters are configured with the same advertising interval, then this may lead to lockstep collision. A small random delay (*AdvDelay*) is added to *AdvInterval* to each broadcast event to avoid this. The value of *AdvDelay* is in the range 0 to 10 ms. If each Broadcasters maintain the fixed delay (τ_{WA}) between the successive advertisements in three channels, then the advertisements on channels 38 and 39 are more likely to succeed when the advertisement on channel 37 is successful. The maximum size of the ADV_IND frame is 47 bytes with 31-byte payload. If there is no collision, this frame is transmitted in 0.376 ms.

An Observer scans the advertising channels 37, 38 and 39, one by one periodically to find the advertised data during *ScanWindow*. The *ScanInterval* is the interval between the start of two successive *ScanWindow* activities. In BLE specification, the *ScanInterval* and *ScanWindow* sizes are limited to a maximum of 10.24 seconds. The *ScanWindow* should be less than or equal to *ScanInterval* as shown in Figure 3.

Fig. 3. BLE Scanning[13]

Two scanning modes are defined: *continuous scanning mode* and *discontinuous scanning mode*. In the *continuous scanning mode*, *ScanInterval* is equal to *ScanWindow*, and the scanner scans each advertising channel without sleeping. On the other hand, in the *discontinuous scanning mode*, the scanner alternately repeats scanning in every *ScanInterval* and sleeps for another period of *ScanInterval*. In the discontinuous scanning mode, *ScanWindow* should be shorter than *ScanInterval*.

There is another independent mode of scanning operation; a scanner can operate in *passive* or *active* mode. In *active* mode, further exchange is supported between the advertiser and the scanner as shown in Figure 4. This exchange leads to connection establishment between two BLEs. In general, the Broadcaster and Observer pair operate in passive mode. There is a mandatory 150μs delay that must be maintained between each packet sent over medium. This is known as the Inter Frame Space (T_{IFS}).

In active scanning, the τ_{WA} should be large enough to support the exchange of ADV_IND, SCAN_REQ, and SCAN_RSP. The BLE specification suggests τ_{WA} value should be less than 10 ms.

Fig. 4. Active Scanning[13]

There are constraints to BLE operations imposed by libraries and smartphones. Android API suggests not more than 5 active scan cycles within 30 seconds, and a preferred duty cycle of about 33.3%. The schedule can be configured to meet these constraints even though illustrations in this paper may have minor digressions to some constraints.

3. PROXIMITY DATA EXCHANGE

The contact tracing application that runs on smartphones includes user facing App and background services for scanning and broadcast. In this paper the term App is used to represent the complete contact tracing application with all its services. This App uses BLE broadcast message, ADV_IND, to exchange proximity data *non-intrusively*. The physical layer frame format of ADV_IND is shown in Figure 5. This 47-byte frame includes advertiser's identification (6-byte address of source BLE), UUID, and payload data. The frame size can be less than or equal to 47-byte depending upon the payload size. The *Access Address* field is used to identify connections uniquely. The

Preamble indicates the start of a new frame and *Cyclic Redundancy Check (CRC)* provides the frame check sequence. The payload independent frame details can be found in BLE specification[9]. The payload size and content are proximity application specific. The description of one such payload can be found in[5]. This payload data contains the Rotating Proximity Identifier (RPID), BLE signal strength at the source, and necessary overhead. All these data items are encoded using tag, length, and value format.

Fig. 5. BLE ADV_IND Frame Format

The heuristic exchange schedule described in this paper is independent of the encoding of proximity data in ADV_IND payload and the recycling of RPID and MAC. However, the encoding is expected to include RPID and the BLE signal strength at the source encoded in some form. The heuristic is also independent of how this data is processed and what type of anonymity is used. A non-intrusive contact tracing App is required to act in the role of both Broadcaster and Observer. The Broadcaster role is implemented by periodic ADV_IND broadcast and the Observer role is implemented by periodic scanning of ADV_IND from other Apps. Each Broadcaster periodically sends ADV_IND with proximity payload in three broadcast channels by maintaining short, fixed delay between these broadcasts. This fixed delay makes it flexible to scan any of the advertising channels for broadcast. The advertisement operation supports Observers which are operating only in passive mode. This means, there is no other interaction between the Observer and the Broadcaster during this exchange (Figure 4). Each Observer also operates in *continuous* mode with a configured duty cycle. It is assumed that a smartphone can perform scanning and advertisement at the same time.

4. OPERATIONAL AND RELIABILITY MODELS

A model is developed in this section for reliable scanning of proximity data with smartphone BLE. This model is then used to develop a heuristic scheduler for proximity service in smartphones. The maximum operational range of BLE is about 330 feet. The effective range could be lower by an order due to various factors like transmitter power, sensitivity of the receiver, and operational environment. Also, there is an issue with the accuracy of the distance computed with BLE differential signal strength[16]. Considering these factors, the operational range r of a smartphone's BLE is assumed to be about 10 feet for the heuristic. This distance is not entirely arbitrary. It is based on the accuracy of distance measurement and the required proximity distance suggested by CDC guidelines. This distance r can be adjusted for different requirements. In an operational scenario, there may be broadcasts from Apps which are beyond this range. This out-of-range data can be filtered out in post processing. The Apps that are outside this range are assumed to have less influence on media contention due to their weaker signal and other factors like environment. The area of a circle with a 10-feet radius is 100π feet². In crowded areas a standing person occupies about 4.5 feet² on the average. This means there can be nearly 70 people within 100π feet². The value of r and the average area occupied by a single person can be altered. These values are chosen to illustrate the heuristic and stay close to realistic deployment requirements.

Fig. 6. Smartphone S1 at the Centre of a Circle with 12 feet Radius (r)

An App S1 and eleven other Apps (S2 to S12) are within mutual data exchange range as shown in Figure 6. The broadcast and scanning schedules of

these 12 Apps are illustrated in Figure 7. Each App is in peer relationship with other Apps in its operational range. Each App is operating in BLE Broadcaster and Observer roles. A slot in the diagram models a unit of time, s . The value of s is fixed, and it is common to all the Apps of proximity data exchange. It is set to 12.5 ms for all our illustrations. The value of s is chosen to be large enough to complete the broadcast attempt within a single slot for all possible values of $AdvDelay$ (< 10 ms). Both scanning and broadcasting are periodic. The scanning period T_s and broadcast period T_b can be expressed as integral multiples of s . The scanning period T_s includes active period T_{sa} and inactive period T_{si} . In the active period, broadcast messages from peer BLEs are scanned and in the inactive period there is no scanning. The time value increases from left to right with column numbers in Figure 7. The S1 row in the diagram illustrates the scanning and broadcast schedules of App S1; the rest of the rows illustrate the scanning schedules for Apps S2 to S12.

	S1 - Scanning & Broadcast (T _b)												S1 - Non-Scanning & Broadcast (T _{si})											
S1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
S2	3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
S3	3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
S4	3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
S5	3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
S6	3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
S7	3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
S8	3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
S9	3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
S10	3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
S11	3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
S12	3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Fig. 7. A broadcast and scanning schedule with 12 Smartphones.

The broadcast period T_b , the active scanning period T_{sa} and inactive scanning period T_{si} are configurable for each App as a multiple of s . The time slots of different Apps are synchronized in Figure 7. However, the heuristic supports Apps with independent clocks. There are 12 Apps in this illustration; the heuristic can support scenarios with more Apps within operational range. The grey slots in S1-row are broadcast slots for S1. That is, S1 sends its ADV_IND message sometime within the duration of the slot. For clarity, broadcast activities of S2 to S12 are not shown in the diagram. A slot

with suffix 'S' in rows S2 to S12 indicates the scanning operation where the corresponding App scans S1's broadcast. The scanning operation includes alternating active (T_{sa}) and inactive periods (T_{si}) with a 50% duty cycle. The inactive cycle time is used to process proximity records from the preceding active cycle and decide if the schedule (T_b , T_{sa} , T_{si}) needs to be modified for the next scanning operation. Each node attempts its broadcast every bs milliseconds. If all the Apps follow the same broadcast and scanning schedules, each App will likely pick up the same number of broadcasts from S1 even when the two clocks are not synchronized. This is illustrated in Figure 7. A trivial way to guarantee this is by using a single common schedule all the time. This trivial solution is not optimal: 1. It may not conserve power at smaller loads 2. It may not offer required reliability for different load conditions. The following section addresses these limitations.

4.1 SCANNING RELIABILITY

A broadcast over a single channel requires about 0.526 ms (transmission time of 0.376 ms + ($\tau_{WA} - 0.376$)). The value of ($\tau_{WA} - 0.376$) should be equal to T_{IFS} (150 μ s). There is no need for it to support SCAN_REQ and SCAN-RESP in this schedule. If there are n Apps within mutual operating range of r feet with uniformly distributed broadcast time, then the probability of success for a broadcast can be computed using one of the generalizations of the Birthday problem[17-19]. The other BLE activities with Central and Peripheral roles are expected to interrupt proximity data exchange operations. They are bursty in nature and assumed to be insignificant considering the perpetual operation of proximity exchange operation. The number of days d is the number of possible broadcasts ($bs/0.526$) within a single broadcast period bs ms. The number of individuals, n , is the number of Apps within the range under consideration. An approximation based on Taylor Series expansion can be applied to compute the probability of successful transmission. This approximation requires d to be larger than n . In our heuristic scheduling the value of d is maintained to be larger than n by an order or more. The probability p that there is no collision for an advertisement is expressed as:

$$= 1 - (1 - e^{-n \times n / 2d})$$

replacing d by $bs/0.526$

$$= e^{-n \times n / 3.8bs}$$

A successful broadcast guarantees successful reception by any actively scanning Observer. The probability of receiving a broadcast is improved when the same advertisement is broadcast multiple times within each scanning period. This can be done by configuring active scanning period T_{sa} as an integral multiple of bs . If each advertisement appears m times within each active scanning period, then the probability R of scanning at least one of these m advertisements within a scanning period is:

$$= 1 - (1 - p)^m$$

This probability is a measure of the reliability of scanning. The value of R is a function of three parameters - broadcast period bs , number of Apps n within the operating range of the Observer, and number of broadcasts m that can be scanned in a single scanning period from a single Broadcaster. The value of m is a measure of active scanning period. The reliability measure can be improved by having large values of broadcast period and active scanning period. There are restrictions to having large scanning and broadcast periods. In general, a scheduler alters the values of T_b (or bs), T_{sa} , and T_{si} for different load conditions to offer better reliability and conserve power.

Max Load (n)	bs = 500, m = 3	bs = 500, m = 4	bs = 1000, m = 2	bs = 1000, m = 3	bs = 1000, m = 4
	Col 1	Col 2	Col 3	Col 4	Col 5
10	1.000	1.000	0.999	1.000	1.000
15	0.999	1.000	0.997	1.000	1.000
20	0.993	0.999	0.990	0.999	1.000
25	0.978	0.994	0.977	0.997	0.999
30	0.947	0.980	0.956	0.991	0.998
40	0.818	0.897	0.883	0.960	0.986
45	0.721	0.818	0.831	0.931	0.971
50	0.612	0.717	0.770	0.890	0.947
55	0.499	0.602	0.701	0.837	0.911
60	0.391	0.484	0.628	0.773	0.862
65	0.295	0.372	0.553	0.701	0.800
70	0.214	0.275	0.478	0.623	0.728

Table 1. Reliability with Broadcast Periods of 500 and 1000 ms

5. SCHEDULING PARAMETERS – T_b , T_{sa} , and T_{si}

The number of days d is equal to $bs/0.526$; where 0.526 is the broadcast time for a single advertisement in a channel. The value of d should be larger than the peak load (70) by an order or more

for Taylor Series approximation. A broadcast period of 500 ms can support a d value of about 950.

The three parameters, namely T_{sa} , b , and m are related. The value of T_{sa} is equal to $(bs)*m$. Controlling two of the three is sufficient to realize the required reliability. A large value of m implies larger T_{sa} . Larger active periods can drain more power. Larger active periods also require larger inactive periods to maintain lower duty cycles. An active period of 4000 ms requires an inactive period of 8000 ms for a duty cycle of 33%. This inactive period is large and may fail to scan some proximity data consistently for the chosen operational range of 10 feet. An inactive period is limited to around 6000 ms in all our illustrations. This limits the broadcast period to around 1000 ms for a duty cycle ratio of 1/3 or less. Rest of the paper evaluates schedules for broadcast periods of 500 to 1000 ms. This choice of broadcast period also limits the value of m to 3 to 6.

Figure 8. Plot of Table 1 Reliability in Y-axis, Load in X-axis

The Table 1 lists reliability values for some broadcast and scanning periods to understand how these two periods influence reliability value. The data is plotted in Figure 8. The reliability values are closely distributed for different schedules at lower load sizes ($n < 40$) and diverge rapidly at larger load sizes. The reliability value also stays higher over a larger range of n for a combination of larger broadcast period and larger value of m . For example, a broadcast period of 1000 ms with an m value of 4 offers better reliability up to a load of 70 (column 5). A smaller value of m supports reduced power requirement. This means, a scheduler should be built with a lower value of m for smaller loads and a larger value of m for larger loads.

Having two schedules, one for smaller load and one for larger load creates a situation with Apps operating with different scanning schedules. This is fine, a scanning schedule chosen by an App has no implication to other Apps. The scanning schedule used is based on the load seen by an App. Apps at the edge with fewer Apps around it can operate with smaller m and the ones that are closer to the centre can operate with larger m . If all the Apps are still using the same broadcast schedule, they can operate with either of the two schedules. Having a common broadcast period across all the Apps makes the reliability computation consistent among all the Apps irrespective of their scanning schedules. All the schedules suggested in this paper assume a single common broadcast period.

	Scheduler 1		Scheduler 2		
T_{sl}	5000	5000	6000	6000	6000
T_{sa}	1000	2000	1200	1800	2400
T_b	500	500	600	600	600
m	2	4	2	3	4
d	947	947	1,136	1,136	1,136
Dcycle	16.7%	28.6%	16.7%	23.1%	28.6%
Load n	1	2	4	5	6
5	0.9998	0.9998	0.9999	1.0000	1.0000
10	0.9974	0.9974	0.9982	0.9999	1.0000
15	0.9877	0.9877	0.9913	0.9992	0.9999
20	0.9644	0.9644	0.9744	0.9959	0.9993
25	0.9223	0.9223	0.9432	0.9865	0.9968
30	0.8591	0.8591	0.8948	0.9659	0.9889
35	0.7764	0.7764	0.8290	0.9293	0.9708
40	0.6788	0.6788	0.7481	0.8735	0.9365
45	0.5735	0.5735	0.6565	0.7987	0.8820
50	0.4681	0.4681	0.5598	0.7079	0.8062
55	0.3691	0.3691	0.4638	0.6073	0.7125
60	0.2814	0.2814	0.3734	0.5040	0.6073
65	0.2076	0.2076	0.2923	0.4046	0.4991
70	0.1484	0.1484	0.2226	0.3145	0.3956

Table 2. Some Possible Schedules with $\leq 33\%$ Duty Cycle

5.1 SCHEDULING WITH 2 OPTIONS

Table 2 illustrates the schedules of two different schedulers with fixed broadcast periods of 500 and 600 ms respectively. These two schedulers operate within the constraints discussed in earlier sections: duty cycle is less than or equal to $1/3$, inactive period is less than 6000 ms, and broadcast period is 500 ms or more.

The scheduler 1 in Table 2 operates with two scheduling options. If the load size is 30 or more, the scheduler uses a longer active period with $m = 4$. If the load size is less than 30, the scheduler switches back to $m = 2$.

5.2 SCHEDULING WITH MULTIPLE OPTIONS

A stable load likely indicates less churn within the operational range. The duty cycle can be reduced to

conserve power when the load is stable at lower ranges. This is done by extending the inactive period within 6000 ms. There is no reduction to scanning reliability, however, the longer inactive period could likely have more misses. This is illustrated in Table 3, the scheduler 1 from Table 2 is made to include one more scheduling option by adjusting the duty cycle without varying broadcast and scanning periods. The column 1 T_{sl} is 1000 ms more than column 2. The inactive scanning period is used to assess this condition and any change in load size. The scheduler in Table 3 shows 4 scheduling options with different values of m . More options can be added with simple addition of multiple acceptable T_{sl} values.

	Scheduler 1				Scheduler 2			
T_{sl}	6000	5000	5000	5000	5000	6000	6000	
T_{sa}	1000	1000	2000	1200	1800	2400	3000	
T_b	500	500	500	600	600	600	600	
m	2	2	4	2	3	4	5	
d	947	947	947	1,136	1,136	1,136	1,136	
Dcycle	14.3%	16.7%	28.6%	19.4%	26.5%	28.6%	33.3%	
Load n	1	2	3	4	5	6	7	
5	0.9998	0.9998	1.0000	0.9999	1.0000	1.0000	1.0000	
10	0.9974	0.9974	1.0000	0.9982	0.9999	1.0000	1.0000	
15	0.9877	0.9877	0.9998	0.9913	0.9992	0.9999	1.0000	
20	0.9644	0.9644	0.9987	0.9744	0.9959	0.9993	0.9999	
25	0.9223	0.9223	0.9940	0.9432	0.9865	0.9968	0.9992	
30	0.8591	0.8591	0.9802	0.8948	0.9659	0.9889	0.9964	
35	0.7764	0.7764	0.9500	0.8290	0.9293	0.9708	0.9879	
40	0.6788	0.6788	0.8968	0.7481	0.8735	0.9365	0.9681	
45	0.5735	0.5735	0.8181	0.6565	0.7987	0.8820	0.9308	
50	0.4681	0.4681	0.7170	0.5598	0.7079	0.8062	0.8714	
55	0.3691	0.3691	0.6019	0.4638	0.6073	0.7125	0.7894	
60	0.2814	0.2814	0.4836	0.3734	0.5040	0.6073	0.6892	
65	0.2076	0.2076	0.3721	0.2923	0.4046	0.4991	0.5786	
70	0.1484	0.1484	0.2748	0.2226	0.3145	0.3956	0.4671	

Table 3. Conserving Power with Duty Cycle Changes

5.3 SCHEDULES FOR LARGER LOADS

One of the issues with scheduler 1 of Table 3 is the low reliability value at larger loads; the value drops below 0.5. Increasing the broadcast period improves the reliability value at larger loads. This fact is illustrated in Table 4 with scheduler 2. It offers more than 0.5 reliability for up to a load of 70 without violating any constraints discussed earlier. This scheduler uses three scheduling options – columns 4, 5 and 6. More options can be added by including lower duty cycles. This scheduler operates with a broadcast period of 1000 ms, and three different scanning periods with $m = 1, 2$, and 3. The duty cycle ratio is less than or equal to $1/3$ and the largest inactive period is 6000 ms.

5.4 SCHEDULING

The scheduling options and their constraints are reviewed in the previous sections. A scheduler

needs to assess the load conditions and choose the appropriate option for its operation within these constraints. This is done during every inactive scanning period. This assessment includes checking the stability of the load and any significant change in load value. If there is a change, then a suitable option is chosen for the next active cycle. This agility is optimal. The default option is the one with the largest load. A scheduler always starts with the default option.

	Scheduler 1			Scheduler 2		
T_{sl}	6000	5000	4000	5000	5000	6000
T_{sa}	1000	1000	2000	1000	2000	3000
T_b	500	500	500	1000	1000	1000
m	2	2	4	1	2	3
d	947	947	947	1,894	1,894	1,894
Dcycle	14.3%	16.7%	33.3%	16.7%	28.6%	33.3%
Load n	Col 1	Col 2	Col 3	Col 4	Col 5	Col 6
5	0.9998	0.9998	1.0000	0.9935	1.0000	1.0000
10	0.9974	0.9974	1.0000	0.9742	0.9993	1.0000
15	0.9877	0.9877	0.9998	0.9429	0.9967	0.9998
20	0.9644	0.9644	0.9987	0.9007	0.9901	0.9990
25	0.9223	0.9223	0.9940	0.8493	0.9773	0.9966
30	0.8591	0.8591	0.9802	0.7904	0.9561	0.9908
35	0.7764	0.7764	0.9500	0.7260	0.9249	0.9794
40	0.6788	0.6788	0.8968	0.6582	0.8832	0.9601
45	0.5735	0.5735	0.8181	0.5890	0.8311	0.9306
50	0.4681	0.4681	0.7170	0.5202	0.7698	0.8896
55	0.3691	0.3691	0.6019	0.4535	0.7014	0.8368
60	0.2814	0.2814	0.4836	0.3902	0.6282	0.7733
65	0.2076	0.2076	0.3721	0.3314	0.5530	0.7012
70	0.1484	0.1484	0.2748	0.2778	0.4785	0.6234

Table 4. Schedulers for Larger Loads

The constraints presented in earlier sections are due to BLE data transmit rate and the chosen operational range, and inactive scanning period. These constraints are likely to change with newer versions of BLE and other requirements. The scheduler presented in this paper can still be easily evolved for these changes.

Same payload size with twice the transmit rate and $T_b = 1000$ ms			Larger payload that requires 0.4 ms for transmit time & $T_b = 1000$ ms		
Load (n)	R m = 2	R m = 3	Load (n)	R m = 2	R m = 3
20	0.996	1.000	20	0.994	1.000
30	0.981	0.997	30	0.973	0.996
40	0.946	0.987	40	0.926	0.980
50	0.885	0.961	50	0.847	0.940
60	0.798	0.910	60	0.739	0.867
70	0.691	0.828	70	0.613	0.759
80	0.573	0.721	80	0.482	0.627
90	0.455	0.597	90	0.360	0.488
a			b		

Tables 5a and 5b. Reliability Measures for Bluetooth 5.0

5.5 SCHEDULING WITH BLUETOOTH 5.0.

BLE supports a single transmission rate of 1 Mbps. The current version of Bluetooth, Bluetooth 5.0, supports wider operational range with four discrete transmission rates: 125 Kbps, 500 Kbps, 1 Mbps, and

2 Mbps. The payload capacity of Bluetooth 5.0 broadcast message is about 8 times that of BLE. If the Bluetooth 5.0 in smartphones is configured to use 2 Mbps transmission rate and continue to use the same proximity payload (31-bytes), then a scheduler could support a larger load with wider operational area (about 12 feet) as illustrated by Table 5a. The same T_{fs} value of 150 μ s is assumed for Bluetooth 5.0. The reliability values for a larger payload that requires the 0.4 ms transmission time are listed in Table 5b.

The graceful migration to Bluetooth 5.0 may require the App to advertise and scan both legacy and new payloads until BLE is phased out. This requires each Bluetooth 5.0 App to send two different broadcast messages one for BLE and another for Bluetooth 5.0 on every advertisement. This could bring down the effective range until BLE is phased out. There are other enhancements to Bluetooth 5.0 with no impact to the heuristic.

5.6 ESTIMATION OF RAW RECORDS

When an App scans an ADV_IND broadcast, it creates a new raw record using the payload, signal strength at the reception, and time stamp. This record is stored and subsequently processed. The number of raw records collected by an App varies with the scanning and broadcast schedules and the number of other Apps in the operating range in each scanning cycle. An App that is operating in an urban setting is likely to collect more raw records than an App that operates in a rural setting. The number of records collected in a day depends on the number of cumulative Apps that were in the operational range during each scanning period of the day. This cumulative value is stochastic in nature. A simple estimation model is built with the average number of Apps in the range for a given scanning period. Assuming an average of 2, there are $2 \times 2 \times 0.9998$ records for every scanning period of 7 seconds (using the data in Table 3). This is about 50,000 raw records for a day. This simple estimate can be used to estimate the server capacity required and how to scale this capacity for fluctuating load conditions.

5.7 COEXISTENCE WITH OTHER BLE SERVICES

The discovery procedure can be shared by all proximity applications including contact tracing.

The BLE in smartphones supports other features besides proximity services. These services can continue to coexist with the proximity services. Occasionally, the scanning and broadcast schedules can be interrupted by the other services employed by BLE Central and Peripheral roles. For instance, when a smartphone is trying to discover an external BLE speaker, it needs to scan on all the 3 advertising channels until the speaker is discovered. Once the speaker is discovered and connected, the contention to advertising channels stops. The Peripheral discovery activities are bursty in nature and are of short duration in relation to perpetually running proximity data collection. Their impact on the proximity service is insignificant. The other assumption is that there are no other perpetually running BLE services that contend with the proximity scheduler.

6. CONCLUSION

A heuristic broadcast and scanning scheduler for proximity applications, more specifically for contact tracing is presented in this paper along with a scanning reliability model. The scheduler operates non-intrusively and it adapts to changing loads with optimal agility. The heuristic uses the built-in BLE in passive Observer and Broadcaster roles. The heuristic is developed for Bluetooth 4.2 (Bluetooth Low Energy) and can be extended to the current Bluetooth version 5.0. This scheduler also provides a basis to estimate the volume of proximity data. This estimate is essential for capacity planning and identifying design issues. The heuristic is simple and developed to work with the limited resources available in smartphones. The scheduler is designed to maximize scanning reliability, minimize Bluetooth power requirements, and adapt to changing load optimally. The scheduler operates within the constraints imposed by BLE driver and smartphone API. The model can evolve as these constraints change.

REFERENCES

- [1] Public Health Guidance for Community-Related Exposure (2020), COVID-19 Guidance, Centers for Disease Control and Prevention, US.
<https://www.cdc.gov/coronavirus/2019-ncov/php/public-health-recommendations.html>
- [2] N. Ahmed *et al.*, "A Survey of COVID-19 Contact Tracing Apps," in *IEEE Access*, vol. 8, pp. 134577-134601, 2020, doi: 10.1109/ACCESS.2020.3010226.
- [3] Contact Tracing App – India (2020), Arogya Setu Official site: <https://www.mygov.in/aarogya-setu-app/>
- [4] Jason Bay et al (2020), BlueTrace: A privacy-preserving protocol for community-driven *Contact Tracing* across borders, A White paper from Government Technology Agency, Singapore
https://TraceTogether.io/static/TraceTogether_whitepaper-938063656596c104632def383eb33b3c.pdf
- [5] Exposure Notification API Preliminary 1.2 (2020), Apple & Google joint Contact Tracing Initiative 2020, <https://covid19.apple.com/contacttracing>
- [6] Exposure Notification Bluetooth Specification 1.2 (2020), Apple & Google joint Contact Tracing Initiative, <https://covid19.apple.com/contacttracing>
- [7] Exposure Notification Cryptography Specification preliminary 1.2, (2020), Apple & Google joint Contact Tracing Initiative, <https://covid19.apple.com/contacttracing>
- [8] Tania Martin, Georgios Karopoulos, José L. Bluetooth Core Specification v5.2 (2019), Bluetooth Core Specification Working Group, Bluetooth SIG Proprietary, <https://www.bluetooth.com/specifications/archived-specifications/>
- [9] M. Afaneh, Bluetooth GATT: How to Design Custom Services & Characteristics [MIDI device use case], Novel Bits, 2017, <https://www.novelbits.io/bluetooth-gatt-services-characteristics/>
- [10] Hernández-Ramos, Georgios Kambourakis,¹ and Igor Nai Fovino, (2020), Demystifying COVID-19 Digital Contact Tracing: A Survey on Frameworks and Mobile Apps, *Wireless Communications and Mobile Computing*, Hindawi, Volume 2020(October), Article ID 8851429, <https://doi.org/10.1155/2020/8851429>
- [11] T.S. Narayanan, Hari (2021): Contact Tracing Solution for Global Community, Manuscript accepted for publication by The Computer Journal, Oxford University Press.
- [12] T S Narayanan, Hari (2021): Novel Distributed and Federated Contact Tracing Systems – A Global Solution. TechRxiv. Preprint.
<https://doi.org/10.36227/techrxiv.14661138.v1>
- [13] Cho, K., et al. 2015, Analysis of latency performance of Bluetooth Low Energy (BLE) networks. *Sensors*, **15**(1), 59–78.
- [14] Cho, K., et al. (2016). Performance analysis of device discovery of Bluetooth Low Energy (BLE) networks. *Computer Communications*, **81**, 72–85.
- [15] Luo, B., Xu, J. & Sun, Z. 2018, Neighbour discovery latency in Bluetooth Low Energy networks. *Wireless Networks* **26**, 1773–1780.

- [16] Locatify, BLE Beacons for Indoor Positioning, Beacon Limitations, <https://locatify.com/blog/ble-beacons-no-bull-beacon-review/>
- [17] Mario Cortina Borja; John Haigh (September 2007). "The Birthday Problem". *Significance*. Royal Statistical Society. **4** (3): 124–127
- [18] Mathis, Frank H. (1991). "A Generalized Birthday Problem". *SIAM Review*. **33** (2): 265–270.
- [19] M. C. Wendl, (2003) Collision Probability Between Sets of Random Variables, *Statistics and Probability Letters* **64**(3), 249–254.