Abstract—In this paper, we propose a simple and practical scheme for optimization the inquiry process in Bluetooth standards in order to achieve fast, reliable and controlled responses. We show how the inquiry process is related to clock setting in Bluetooth devices and how clock optimization could be used to shorten the inquiry process time. Theoretical analysis and simulation results shows the efficiency of the proposed optimized scheme. We also develop a traffic transportation application, i.e. speed measurement, as an example of many applications where this scheme could benefit. Results show how the proposed optimized scheme outperforms in comparison with others.

I. INTRODUCTION

Bluetooth [1] is a wireless communication standard utilizing the 2.4-GHz radio spectrum that has achieved global acceptance. Bluetooth technology exists in many devices such as cell phones, game consoles, headsets, in-vehicle Bluetooth hub, and recently in high-tech watches. Unlike the other wireless standards (e.g., WiFi or Zigbee), Bluetooth standards support the discovery of surrounding nodes. More specifically, Bluetooth standards specify an inquiry process, which a Bluetooth-enabled device (i.e., master device) uses to discover all the other Bluetooth-enabled devices (i.e., slave devices) within the proximity. In this inquiry process, the master device broadcasts an inquiry message in which any slave device receiving this message enters the inquiry response mode and replies with an inquiry response message that contains, at least, the unique 48-bit identifier of the slave device. This inquiry process allows detecting and tracking the individual devices in the vicinity. Because of this interesting built-in device discovery feature, many researchers and industries try to utilize the Bluetooth technology in several applications and services.

One of the areas in which Bluetooth can be exploited is transportation. Bluetooth now as a built-in capability in vehicles exists for hands-free calls as well as for transmitting of digital music files to a vehicle’s stereo system. It is expected that by 2016, about 70% of vehicles will be equipped with built-in Bluetooth [2]. In spite of the widespread of Bluetooth devices in vehicles, Bluetooth technology has been utilized in transportation domain only in highways scenarios [3]. We envision to utilize the Bluetooth technology in building sensing platform to collect traffic data such as the actual number of vehicles, vehicles’ speed, vehicle’s positions, queue length and lane blockages through tracking vehicles at urban signalized intersections and streets. This Bluetooth sensing platform has many advantages including: a) No need to equip or prepare the vehicles with any additional component, b) No need for special-purpose sensors (e.g. pneumatic road tubes, magnetic loops) at roadside, c) very simple and fast deployment and maintenance of Bluetooth receivers at roadside since there is no need to tear up any piece of the road, and d) The cost of integrating Bluetooth receivers with traffic road-side infrastructure (e.g., traffic controllers) is minimized. This platform will enable the development of new interesting applications and services for transportation field.

Several challenges need to be addressed for Bluetooth exploitation in transportation applications. Bluetooth standards [1] are designed for a few stationary Bluetooth nodes. On the contrary, transportation applications are dealing with high speed mobile nodes in high density scenarios (i.e., vehicles with Bluetooth capability). Therefore, in order to utilize Bluetooth in transportation applications, Bluetooth standards need to be adjusted and tuned to this new characteristics. More specifically, the Bluetooth inquiry process needs enhancements to enable the detection of as many vehicles as possible in the shortest possible time. Typical inquiry process, as we describe later, could take up to 10 seconds to discover a device. This performance is not acceptable in the high dynamic transportation scenarios. For example, in a typical urban scenarios with average speed of 35 miles/hour (16 m/s), given the typical Bluetooth coverage range of 10m, a slave has only 1.25s to receive an inquiry request from a master and response with the inquiry response in order to be detected. Given the typical behavior of Bluetooth slave in which it listens to receive an inquiry packet once every 1.28s, the 1.25s is enough for only one attempt by the master to try to discover the slave. This becomes more challenging with shorter Bluetooth ranges as it is required for urban scenarios, or in scenarios with higher speeds. Moreover, in a high dense scenario of vehicles, which is typical in urban environment, collisions and consequently more delays could be expected. Therefore, designing an optimized inquiry process is a must to meet the new requirements of transportation services and applications.

Many methods have been proposed to reduce the inquiry process time of Bluetooth, e.g. [4], [5]. Some of these methods such as [5] requires modifications in the slaves, which is not practical for transportation applications. Our focus in this paper is on applying simple and practical modifications to the Bluetooth masters only (i.e., infrastructure side) to enhance the inquiry process. More specifically, in this paper, we optimize and evaluate the performance of the inquiry process through clock modifications when multiple Bluetooth masters are used in a transparent way to the slaves. We show how much improvement can be achieved both theoretically and practically to enhance the inquiry process both in speed and accuracy. Performance evaluations show that the proposed
scheme performs near optimal theoretical values.

The rest of the paper is organized as follows. Section II gives a brief overview of the Bluetooth inquiry process. Our discussion follows in Section III with describing a probabilistic model of inquiry process between masters and slaves. We analyze the effect of having multiple masters. We show how a scheme can be developed to achieve the best average inquiry time via clock manipulating. Simulation results of schemes are presented in Section IV. A simple application that uses these schemes is developed and evaluated in Section V. Finally, the paper is concluded in Section VII.

II. BLUETOOTH INQUIRY PROCESS

A Bluetooth device hops over 79 defined frequencies in Bluetooth standard. Detail of frequency hopping can be found at [1]. For inquiry process, which we focus on in this paper, the main parameters that affect the hopping sequence are native clock of Bluetooth device (i.e., a 28 bits counter that increments in 312.5 µs time unit), $N$ (a 5 bits counter that is initialized randomly) and $k_{\text{offset}}$ (an integer with a value of either 8 or 24). The role of these parameters in the inquiry process will be described later in this section.

The inquiry process\(^1\) happens as follows. In each slot (two pulses of 312.5 µs), an inquiry packet (ID) is sent by the master. In the next slot, any slave that has received the previous inquiry, responds with an inquiry response packet (FHS). The slave should be in inquiry scan substate in order to listen to the inquires. The slave enters inquiry response substate to send the FHS that contains the native clock of the slave and the lower portion of slave’s Bluetooth address, (LAP), among others. Because ID packets have short length, it is possible to send two ID packets in one slot. However, if two slaves receive those two ID packets and response to it, the master can only receive one of them because FHS packet is long and needs a duration of one slot to be received. In inquiry substate, the master will hop over a sequence of 16 channels, called train, in a duration of 10 ms. These 16 channels are selected by sequence selection scheme for the master through the native (local) clock of master, address (a constant value for inquiry), and $k_{\text{offset}}$ values. The selection of $k_{\text{offset}}$ result in two set of trains that are called A-Train and B-Train, where 24 (for A-Train) or 8 (for B-Train) is selected. Each train is repeated $N_{\text{inquiry}}$ times. During inquiry interval, a sequence of $N_{\text{inquiry}}$ A-trains, $N_{\text{inquiry}}$ B-trains, $N_{\text{inquiry}}$ A-trains, and $N_{\text{inquiry}}$ B-trains of ID packets will be sequentially transmitted, where $N_{\text{inquiry}} = 256$.

Hence, the inquiry process for the master will take $(4 \times 256)$ of A/B-Train, each of 10 ms) 10.24 seconds. The master can finish the inquiry earlier if it receives enough responses or if it finds its target device. The Bluetooth specification suggests that the master enters the inquiry substate every one minute. However, in this study we assume that masters are only in inquiry substate continuously.

\(^1\)We describe the common version of the inquiry process that is mandatory and supported by all Bluetooth versions (i.e., version 1.2 and upper), for the sake of maintaining compatibility.

The slaves enters Inquiry Scan substate to listen for inquiries. The inquiry scan substate lasts in one channel for the length of Inquiry Scan Window which is at least 11.25 ms, or 18 slots. Then, the slave switches to other states and after Inquiry Scan Interval, it returns back to inquiry scan substate again. According to the specification, slaves can take one of these three values for Inquiry Scan Interval: less than 1.28, 1.28 or 2.56 seconds.

Any slave that receives an ID packet on one channel, say $c$, will send its FHS packet in the next slot on the corresponding channel, say $c + 1$. To avoid clock synchronization between slaves that results in collisions, the slave will run a random back-off timer after sending FHS packet and before returning back to inquiry scan substate again. Also, after the transmission of FHS, the parameter $N$ is incremented by the slave. The initial value of $N$ is selected randomly by the slave.

III. INQUIRY PROCESS - ANALYSIS AND OPTIMIZATION

In this section, we evaluate and analyze the optimization of inquiry process via clock manipulation. Since the used address in generating hopping sequence for the inquiry process is constant (i.e., the GIAC address, $0x9E3B33$), it is possible to allow the fast detection of the slaves as soon as they enter their inquiry scan substate only via manipulating the clock values of the masters. Given the importance of knowing the expected delay for discovering the slaves, we discuss the delay values and how we could enhance the inquiry process for faster response.

A. Inquiry Response Delay

In this subsection, we discuss the possible delay values for receiving inquiry responses from the slaves. Ideally, the minimum delay to receive an inquiry response from a slave is 1.25 ms. This minimum delay value occurs when the slave enters the inquiry scan substate at the same time when a master sends the inquiry on the same channel. In this case, the slave will send its response immediately at the next slot. Therefore, the delay in receiving the inquiry response in this ideal scenario is only two slots, i.e. 1.25 ms. On the contrary, the maximum delay for the worst case scenario in an error-free environment, according to Bluetooth specification [1], is 10.24 s. Peterson et al. [6] shows that in 99% of scenarios, a slave could be detected in 5.12 s. In general, the average delay to detect all slaves will depend on the percentage of detected slaves at different time intervals and the length of interval as shown here:

$$\text{Delay}_{\text{avg}} = \frac{\sum_{i=1}^{I} t_i \cdot P_i(t)}{I},$$

where $I$ is number of time intervals, $t_i$ is the mean of $i$th time interval and $P_i(t)$ is the percentage of the slaves that are detected in the $i$th interval. A detailed analysis of the detection percentage $P_i(t)$ is out of the scope of this paper, and could be found in other work, e.g., [6]. However, we are interested in developing methods that could detect all inquiry responses within only one interval. We fix the length of the interval to be equal to the Inquiry Scan Interval because it is guaranteed
that all the slaves will enter the inquiry scan state at least once during this interval. For the best case when all slaves are detected in the first interval where inquiry scan interval duration is 1.28s, the mean delay simply is about 646 ms. Table 1 summarizes the range of delay values for the inquiry responses.

<table>
<thead>
<tr>
<th>Min</th>
<th>Best Mean</th>
<th>Mean</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.25 ms</td>
<td>646 ms</td>
<td>Eq. (1)</td>
<td>10.24 s [1] and 99% in 5.12 s [6]</td>
</tr>
</tbody>
</table>

B. Basic Enhancements for Fast Inquiry Response

In this subsection, we describe possible solutions to enhance inquiry process and compare them with each other.

1) Naïve solution: The straightforward solution is to have two masters per each possible channel. One master will be sending ID frames on this channel while the other master is listening to any response on this channel. In this solution, the discovery delay is the min value showed in Table I. However, this solution requires a significant number of masters (roughly 79 · 2 = 158 masters) that make it impractical in terms of cost and management.

2) Frequency Prediction solution: The second solution is to predict the hopping sequence of the slaves. In this solution, after receiving the first inquiry response from a slave, the master will be able to predict the next hopping channel which the slave will switch to for listening to it. Consequently, the master switches to this channel to send the inquiry packet. In the next slot, master will receive the corresponding inquiry response. To predict the frequency hopping of slave for the inquiry scan, the native clock, the value of N, and random backoff value of the slave are required. The native clock of the slave could be detected from the FHS packet. Given that, there is still some uncertainty about the next hopping sequence since the value of N and the random backoff parameter are needed. The value of N could be detected with the reception of multiple inquiry responses since it is an incremental parameter. Although backoff value is chosen randomly, the prediction can be started with the assumption of minimum value, 127 slots. However, this solution is hard and inefficient to be implemented for our purposes since it requires that the master detects few inquiry responses from the slave before starting to predict the next hopping channel. This is not acceptable in the transportation domain since the duration of the Bluetooth coverage would only be enough for the reception of a single inquiry response. Furthermore, the master needs to stay in the predicted channel for a long time period since it does not know the actual value of backoff timer. During this time period, the master keep sending multiple inquiries until it receives a corresponding response. So, almost one master per each slave is required, which is not an efficient solution.

3) Train Improved solution: The third solution is to try to produce different hopping sequence by several masters. The simplest solution is to set two masters to operate with two different trains. However, because of the variation in the selected clock value, there is no guarantee to have two non-overlapping hopping sequences. To extend this scheme to more than two masters, the A-train and B-train could be divided into halves as well. For example, with four masters, we will have four different hopping sequences: A1-train, A2-train, B1-train, and B2-train. Each of A1-train and A2-train will have 8 channels from the 16 channel of A-Train. Similarly, B-train is divided to two sub-trains. This method does not improve the performance of inquiry response significantly. Indeed, dividing trains will only improve the inquiry scan rate for the same set of channels, not increasing the diversity of scanned channels. This leads us to the idea of manipulating the master clocks to maximize the diversity of the hopping channels among multiple masters. This scheme is described in Section III-C.

C. Clock Optimization Enhancement Scheme

As discussed earlier, the native clock is the main parameter in generating the hopping sequence for the inquiry process. Ideally, generating hopping sequences by multiple masters needs to be designed in a way to minimize simultaneous common inquiry channels among them. Toward this aim, the clock values of the masters need to be carefully selected. This is simple for the case of having two masters in which both could use the same clock value while adjusting the \( h_{offset} \) values of the two masters to generate non-overlapping hopping sequences: A-Train and B-Train. However, for a higher number of masters, this setting could result in hopping sequences with several overlapping channels. Hopping sequences with minimum overlapped set of channels will guarantee to generate inquiries over more number of channels and hence will increase the chance of receiving responses faster.

The objective of our proposed scheme is to find the best clock values for \( m \) masters that will generate \( m \) hopping sequences with minimum overlapping channels. In this regard, we define the similarity (distance) metric \( D \) for any two clock values \( i \) and \( j \) in \( S = \{1, 2, \ldots, 2^{28} - 1\} \) as follows:

\[
D(i, j) = q, \quad \forall i, j \in S,
\]

where \( q \) is the number of similar frequencies in both hopping sequences each of length \( l \). For example, Table II shows two hopping sequences of length 8 with distance \( D = 3 \) (frequencies 02, 64 and 45 are similar in both sequences).

<table>
<thead>
<tr>
<th>TABLE II</th>
</tr>
</thead>
<tbody>
<tr>
<td>TWO SAMPLE HOPPING SEQUENCE</td>
</tr>
<tr>
<td>45</td>
</tr>
<tr>
<td>39</td>
</tr>
</tbody>
</table>

Note that the distance metric depends on the length of the hopping sequence, i.e., \( l \). After the hopping sequence of length \( l \) is over, the clock values need to be reinitialized to repeat the same hopping sequences. The extreme case occurs when this duration is equal to one Bluetooth slot in which the clock values are reinitialized every slot. However, the reinitialization period and the corresponding hopping sequence length should be selected in a way which minimizes the overhead of the masters’ synchronization. Since the objective of this paper is to discover as many slaves as possible within the period of Inquiry Scan Interval, we choose the reinitialization period to
be equal to Inquiry Scan Interval, in which the corresponding channel sequence is 128 hops length. Hence, to find the optimum configuration of clock setting, we need to find a set of clock values \(s_1, s_2, \ldots, s_m\) for the \(m\) masters from the set \(S\) that have the minimum number of the overlapped channels, that is given by:

\[
\min_{s_1, s_2, \ldots, s_m \in S} \sum_{i=s_1}^{s_m-1} \sum_{j=i+1}^{s_m} (D(i,j)), i \neq j. \tag{3}
\]

In other words, if we model the generated hopping sequence for each clock value as a node in a bidirectional weighted graph where the weight of the edge between nodes \(i\) and \(j\) is the \(D(i,j)\) from (2), then the objective is to find a set of \(m\) nodes with the minimum sum of their weights. The adjacency matrix of a simple graph of five nodes is shown in the following:

\[
\begin{pmatrix}
1 & 2 & 3 & 4 & 5 \\
1 & 16 & 2 & 3 & 16 & 7 \\
2 & 2 & 16 & 0 & 4 & 2 \\
3 & 3 & 0 & 16 & 12 & 3 \\
4 & 16 & 4 & 12 & 16 & 9 \\
5 & 7 & 2 & 3 & 9 & 16
\end{pmatrix}
\]

In this simple example, if it is required to find three nodes (hopping sequences) with the minimum distance (overlapped channels), then the solution is nodes 1, 2, and 3 where the corresponding sum of distances is \(2 + 3 + 0\). This solution is the minimum compared to any other choice of three nodes.

Our problem is the inverse of edge-weighted maximal clique problem [7] problem in which we are looking for the minimal weighted clique in a complete graph, (vertices could be connected with weighted edges equal to 0). This clique problem is NP-complete and it is modeled typically as 0-1 integer programming optimization problem [8]. Among all meta-heuristic methods for solving the problem, we select Tabu search [9] because of its simple implementation and flexibility for customization to our problem properties. Moreover, by knowing all the weights of similarity matrix, the minimum possible solution for \(m\) master is computed simply and is used as the stop condition of search. We developed a customized tabu search to enhance the stop condition as well as the random search over total search space to avoid local optimum points. We solved the problem when the number of masters, \(m\), vary between 2 and 32 and for 17 bits length clock values\(^2\). The optimum solution will be used to set the clock values of masters in our simulation, which will be described in Section IV.

D. Clock Synchronization

In Section III-B3 and III-C, it is assumed that there is a synchronization mechanism among masters. In fact, schemes that operate based on the clock of masters, need synchronization in order to make sure that the designed clock setting will occur at the same time for all masters. One of many clock synchronization methods or protocols can be used. For example, if the masters are operating on the same platform, clock drift would not be significant even though trivial methods such as Cristian [10] can be used. In other scenarios, where the masters are connected through a network, one of protocols such as Precision Time Protocol (PTP) [11] could be deployed. PTP’s accuracy on order of sub-microseconds has been confirmed [12] that is a satisfactory threshold for Bluetooth’s maximum acceptable clock drift that is 40 \(\mu\)s.

IV. PERFORMANCE EVALUATION

In this section, we present an extensive simulation-based study on the performance of our proposed scheme to optimize Bluetooth inquiry process under different scenarios.

A. Simulation Configuration

The ns-2 simulator [13] with a Bluetooth extension [14] is used for performance evaluations. In our scenarios, Bluetooth nodes are spreaded randomly over 50m \(\times\) 50m area. In this configuration, communication range between any Bluetooth nodes will be limited within 85m that is the typical communication range for industrial Bluetooth devices [15]. We use the BlueHoc interference model from the Bluetooth extension module. We set the duration of each simulation scenario to 15 seconds in which Bluetooth masters are configured to continuously run the inquiry process during the whole simulation period. For Bluetooth slaves, we used the typical Bluetooth configuration, i.e. 1.28s for Inquiry Scan Interval and 11.25ms for Inquiry Scan Window.

Three different Bluetooth inquiry schemes are evaluated under each simulation configuration:

1) **Regular** scheme: the standard inquiry process where clock values are selected randomly for each master.
2) **Improved** scheme: an improved scheme over the Regular scheme where the clock values are still configured randomly but the masters are divided to two groups where one group uses A-train frequencies and the other group uses B-train frequencies.
3) **Optimized** scheme: this is our proposed scheme in which masters clocks are configured as described in Section III-C, which guarantee to generate the optimum frequency hops for the set of \(m\) masters.

We define the following metrics for evaluation:

- **Detection Percentage**: the percentage of detected slaves by at least one master within a specific duration period. We use multiples of Inquiry Scan Interval as different duration periods. Optimally, we are interested to discover as much slaves as possible within the first Inquiry Scan Interval period.
- **Average Detection Delay**: the average time to discover all the slaves since the beginning of the inquiry process.

B. Simulation Results

In this section, the results of simulation for various number of masters and slaves are presented. Results are the average
over 250 runs for each scenario. Figure 1 shows the basic scenario corresponding to one Regular master and one slave. As shown, almost 50% of the times the slave is detected within the period of the first Inquiry scan interval. This result validates that a slave will be detected within the first inquiry scan interval if it is scanning a channel which is one of the hopping channels of the A-train of the master. On the other hand, the slave needs to wait until the master switch to B-train of hopping channels to be discovered. Recall that based on Bluetooth standard, the master switches between trains every 2.56 seconds. For this scenario, the average detection delay is 1.8 s that is aligned with the calculations in (1).

![Graph showing detection percentage using 1 master and 1 slave](image1.png)

**Fig. 1.** Detection Percentage using 1 master and 1 slave

Regular scheme when 2 and 4 master are used, respectively. The performance of these three schemes converge with using large number of masters. This figure confirms the analysis in (1) that the best achievable delay is about half of inquiry scan interval. However, in this scenario, there is only one slave to be detected. As the number of slaves increases, the performance of Regular and Improved schemes drops significantly in comparison to Optimized scheme, both in detection percentage and average detection delay.

Figure 4 shows detection percentage for the case of four masters with various number of slaves. The figure shows that the Optimized scheme detects more than 94% of slaves within the first inquiry scan interval even in a very high dense scenario (e.g., 256 slaves) while regular and improved schemes can only detect 87% of slaves. The average detection delay with four masters in the case of the Optimized scheme is about 790 ms while for regular and improved is about 1 second.

![Graph showing detection percentage using multiple masters and slaves](image3.png)

**Fig. 3.** Average Detection Delay using multiple masters and 1 slave

![Graph showing detection percentage of different schemes](image2.png)

**Fig. 2.** Detection Percentage of Different Schemes using 2 masters and 1 slave

Given only one master is considered in this scenario, Improved and Optimized schemes cannot be used. Figure 2 shows the effect of adding an additional master on the detection percentage of one slave within the first inquiry scan interval for all the three different schemes. Optimized scheme significantly outperforms the other two schemes, in which it has detected 99% of the slave. The corresponding average detection times for the Regular, Improved, and Optimized schemes are 1.25, 1.13 and 0.64 seconds respectively. It is clear from the results that the hopping sequences generated by the Optimized scheme have more number of non-overlapping channels than the hopping sequences generated by the Improved scheme.

Average detection delay is shown in Figure 3 when more masters are used. The Optimized scheme improves the detection delay with up to 53% and 18% in comparison to the

![Graph showing detection percentage using four masters and multiple slaves](image4.png)

**Fig. 4.** Detection Percentage using four masters and multiple slaves

**V. TRANSPORTATION APPLICATION: SPEED MEASUREMENT**

Estimating vehicles’ speed on roads is one of the main applications in transportation systems. In this section, we show that our proposed optimization for inquiry procedure enhances the vehicle’s speed estimation. Figure 5 shows the topology of Bluetooth masters along a road segment to detect passing vehicles in order to estimate vehicle’s speed over this segment.
The measurement application could have wired or wireless connection to Bluetooth masters to retrieve the information about received inquiry responses. Masters at each point will have a control unit that will coordinate clock synchronization among masters, which is required to set the designed clock values for Improved and Optimized schemes. The unit will send commands to Bluetooth devices to set the clock values in specified intervals simultaneously.

![Fig. 5. Topology of Bluetooth Masters](image)

Assume that \( t_1 \) and \( t_2 \) corresponds to the two received inquiry responses at the center of each of the observation points. Then, the speed could be estimated simply as:

\[
S_{est} = \frac{d}{t_2 - t_1},
\]

where \( \delta t \) could be any value from the range \([-2R, 2R]\). The value of \( \delta t \) is due to uncertain delay in detecting the vehicle once it enters the communication range, where \( Max_{\tau} \) is Inquiry Scan Interval with the assumption that the vehicle always gets detected. One way to reduce the uncertainty in this equation is to reduce the communication range \( R \). However, reducing the transmission range increases the possibility of missing the inquiry response. The other alternative to reduce uncertainty is to use a faster inquiry process. This way, the \( \delta t \) error would be minimized. As discussed and shown in previous sections, the Optimized scheme minimize the detection delay. Hence, using the the optimal clock setting in masters will improve the detection and accuracy of the speed measurement. More specifically, our proposed optimized scheme is expected to increase the accuracy of speed measurement even with large communication ranges.

In the following, we evaluate the performance of Optimized scheme in comparison to the regular clock setting of masters for the speed estimation application.

A. Experiment-A

We use our simulation environment in this experiment. We specify two points of observation that are \( d = 100m \) away from each other. We use 1, 2, and 4 masters at each of the observation points with communication range of 13 m radius. Masters are configured to run inquiry scan continuously. Slaves (here vehicles) pass the two observation points. Although we experimented with different vehicle speeds, we present the results corresponding to the speed of 20m/s. Although this is a high speed for urban scenario, we choose it to show that even in worst case, our scheme can estimate speed with an acceptable accuracy. Scenarios are evaluated using Regular, Improved, and Optimized schemes. Time difference between received inquiry responses at each of the observation points will specify our estimated speed.

Figure 6 shows the Success Percentage metric that is the percentage of vehicles that were detected at both observation points. From the figure, it is clearly obvious that the Optimized scheme with 99% detection percentage significantly outperforms the other two schemes in detecting a higher number of vehicles. In fact, the improvements in detection percentage because of using the Optimized scheme is up to 76% and 25% for the cases of 2 and 4 masters respectively. From the experiment, we found that the mean absolute error (MAE) of speed estimation is 0.5 m/s while the standard deviation is 0.1 m/s.

B. Experiment-B

In this experiment, we increase the communication range to increase the chance of vehicles detection. The range is set proportional to the number of masters and schemes to ensure Success Percentage \( \geq 99\% \). Figure 7 shows the MAE for different schemes and various speed values, in which the Optimized scheme outperforms the other two schemes even in high speed scenarios. From the figure, it is clear that Optimized scheme with 4 masters outperforms all other schemes. Even with 2 masters and applying the Optimized scheme, the error is almost 2.3 m/s, which is about 11% for the high speed scenario of 20 m/s. The Optimized scheme with four masters can have 2 and 5 times less error in speed estimation than the Improved and Regular schemes, respectively.

VI. RELATED WORK

There are some limited efforts to improve the performance of inquiry procedure of Bluetooth. For example, Jiang et al. [4] propose reducing the Inquiry Interval to half. So, instead of inquiry every 60 seconds interval, master go to inquiry every 30 seconds. This method is only useful where masters are not allowed to inquiry all the time. However, in our problem, we
set the masters to the inquiry state all the time. Moreover, they evaluate the Dual Inquiry Scan (DIS), in which slaves scan two channels in one scan interval of 1.28 seconds. However, this method requires modifying the slaves that is not practical for real applications. Welsh et al. [5] suggest removing the random backoff timer to reduce the reception time of next responses, which is not useful for our purpose, in which the reception time of the first response should be minimized. This method also requires the slaves modification.

To collect the road’s data traffic, many methods such as pneumatic road tubes, magnetic loops, microwave radars, and video image detection have been proposed in literature [16]. Apart from advantages and disadvantages of those methods in terms of ability to detect all type of data, accuracy, lifetime, and ability to work in various weather condition, most of them are expensive and difficult to deploy [16]. Therefore, recently the use of wireless networks for traffic data collection has attracted attentions due to its ability to work in any condition, relatively low cost and simple setup. Speed and travel time measurements by Bluetooth is a sample application. Puckett et al. [3] show how effectively they can use Bluetooth for speed and travel time measurements in highways and roads where speed is estimated over road segments of kilometers scale. However, they have not considered the urban scenario which missing only one inquiry response can significantly affect the accuracy of collected data, where traffic data needs to be collected on order of meters with a finer granularity of accuracy. Bluetooth has also been used in localization applications. Figueri et al. [17] consider using multiple masters for location estimation in indoor scenarios. They show that the earliest inquiry response reception is a need for localization applications. They, however, focus on using the RSSI values received from inquiry responses for localization. Our methods, which minimize the inquiry responses, can improve such localization methods.

VII. CONCLUSION & FUTURE WORK

In this paper, we investigated the inquiry process in Bluetooth specification. We discussed theoretically how the typical inquiry process, which could take up to 10 seconds to receive an inquiry response from a slave, could be optimized. We proposed new schemes to optimize the delay of inquiry process through manipulation of clock values of masters. We confirmed the outperform of the proposed schemes through simulation. Finally, we showed how these schemes could be utilized in traffic transportation applications such as vehicles speed measurement in urban scenarios.

As future work, we are planning to evaluate our schemes using real testbed. To this aim, the firmware of any Bluetooth chipset has to be modified. Currently, there is no open source Bluetooth firmware available for research community. Therefore, we are exploiting the Ubertooth devices [18] for this purpose. The Ubertooth devices have been designed for sniffing the Bluetooth packets. However, there is not a complete Bluetooth protocol stack implementation. The hardware specifications of Ubertooth devices allow the implementation of Bluetooth protocol stack. Moreover, it is possible to reset the clock values for the ARM Cortex-M3 microcontroller.

We plan on modifying this device in order to implement our proposed schemes and evaluate them using real experiments.

REFERENCES