BlueTorrent: Cooperative Content Sharing for Bluetooth Users*

Sewook Jung, Uichin Lee, Alexander Chang, Dae-Ki Cho†, Mario Gerla
Department of Computer Science
University of California, Los Angeles
{sewookj,uclee,acmchang,gerla}@cs.ucla.edu,†pinecho@ucla.edu

Abstract

People wish to enjoy their everyday lives in various ways, among which entertainment plays a major role. In order to improve lifestyle with more ready access to entertainment content, we propose BlueTorrent, a P2P file sharing application based on ubiquitous Bluetooth-enabled devices such as PDAs, cellphones and smart phones. Using BlueTorrent, people can share audio/video contents as they move about shopping malls, airports, subway stations etc. BlueTorrent poses new challenges caused by limited bandwidth, short communications range, mobile users and variable population density. A key ingredient is efficient peer discovery. This paper approaches the problem by analyzing the Bluetooth periodic inquiry mode and by finding the optimum inquiry/connection time settings. At the application layer, the BlueTorrent index/block dissemination protocol is then designed and analyzed. The entire system is integrated and implemented both in simulation and in an experimental testbed. Simulation and measurement results are used to evaluate and validate the performance of BlueTorrent in content sharing scenarios.

1 Introduction

Bluetooth is an always-on, low-power, and short-range hookup for implementing Wireless Personal Area Networks (WPANs). It became the most popular PAN communication device. The Economists [9] reported that sales of Bluetooth devices are more than doubled in 2005 to reach 320M units, and the figure is expected to exceed 520M this year.

The concept behind Bluetooth is to support universal short-range wireless capabilities using the 2.4GHz globally unlicensed low power band. Bluetooth allows users to connect up to eight devices by forming a star-shaped cluster, called piconet. The cluster head is called master and the other nodes are called slaves. Two Bluetooth devices within 10 m range can exchange data up to 723kbps. To minimize interference among different Piconets, Bluetooth uses frequency hopping (FH) with pseudo-random ordering of 79 frequencies in the same band.

Bluetooth is intended to various applications, including audio and data. As widely observed these days, Bluetooth gradually replaces cables to link computers to keyboards, printers, and mouse. It is often treated as a multimedia enabler such that users can listen to and download music from other machines using wireless headsets and MP3 players. More importantly, new legislation banning the use of mobile phones without a hands-free kit while driving boosted the use of Bluetooth-based mobile headsets in public. This adoption has in turn cleared the way for the inclusion of Bluetooth in all kinds of new products. For example, automotive manufacturers are looking to link Bluetooth devices with the on-board system to provide in-car stereo/hands-free solution.

With Bluetooth's continuing proliferation, Bluetooth-based applications are well positioned to deliver new opportunities to all facets of the industry. This paper deals with one fast growing application – namely, proximity advertising and marketing. In 2005, handset maker Nokia and music label EMI started a project to let coffee shop customers listen to music via Bluetooth [2]. Similarly, San Francisco's WideRay has placed Bluetooth-enabled kiosks in selected music stores, video game stores, and theaters across the country to send messages asking if users are interested in getting more information about various items the store is selling, e.g., music, ring tones, video etc. More interestingly, in August 2006 CBS announced that it will make clips from its prime-time fall program line-up available for free via CBS Outdoor billboards [6]. The digital billboards, initially located in NYC's Grand Central Station, will allow cellphone or PDA users to watch clips such as CSI. There are already off-the-shelf commercial products available for

*This work was supported in part by the National Science Foundation under Grant No. 0520332 and the US Army under MURI award W911NF-05-1-0246. The research is continuing through participation in the International Technology Alliance sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-06-3-0001.

proximity marketing such as BlueBlitz and BlueCasting.

However, proximity marketing solutions so far are based on a client and server model. A service provider spreads advertisements using a Bluetooth Access Point (BT-AP) to mobile hosts within Bluetooth communication range (i.e., 10m). However, since users are mobile and the Bluetooth channel is error-prone, the total amount of data that a mobile user can download from the server is limited to a few hundreds kilobytes – i.e., only images or small video clips. Moreover, it is not practical nor economically feasible to install BT-APs every 10m. Under such circumstances, distributing contents larger than several hundred KB requires customers to stop in front of the BT-AP, unless we adopt a P2P technology.

In this paper, we propose BlueTorrent, a cooperative content sharing protocol for Bluetooth that exploits sparsely distributed BT-APs allowing mobile users to cooperatively download relatively large files (e.g., 1-10Mbytes). The challenge now becomes to support P2P connections among mobile Bluetooth users. In Bluetooth, one node must be a master and the other node must be a slave: the role must be dynamically changed with a frequency that minimizes the discovery latency, yet allowing enough time for useful peer to peer data transfers. To this end, we analyze an existing standard function and find the optimal parameter configurations. To effectively share content in spite of short link duration, BlueTorrent uses BitTorrent-like file swarming. Content is divided into a number of small pieces, and mobile users can exchange whatever pieces are available. BlueTorrent uses a cooperative carry and forward strategy: pieces are forwarded whenever a connection is available in order to minimize download latency. Note that meta-data (e.g., unique file ID, title, media type) of a file is also spread opportunistically via mobility. BlueTorrent users can actively query other peers to search for a file of interests. The following is the key results of our study:

- We find that our simple periodical inquiry scheme at the application layer results in better performance than the standard inquiry mode. The latter only supports parameter control at the time scale of seconds, thus resulting in performance degradation.

- We identify key parameters and their configurations to minimize the peer discovery latency. Among them, we show that the inquiry scan period plays a key role in determining the optimal configuration of the other parameters.

- We validate the performance of BlueTorrent via simulations and testbed experiments. For some tested scenario, our cooperative data sharing results in more than 400% improvement in terms of download latency. We show that the size of a piece is important in mobile content sharing. For AP mode, we find that for a given user density, there exists the optimal speed, leading the best performance.

This paper is organized as follows. In Section 2, we explain Bluetooth. In Section 3, we analyze the periodic inquiry mode in Bluetooth for P2P connections and find optimal configurations. In Section 4, we propose and analyze our index and content sharing protocol. In Section 5, we evaluate our protocols through extensive simulations. In Section 6, we show the preliminary performance results of BlueTorrent in our testbed. In Section 7, we review the related work and then conclude the paper in Section 8.

2 Bluetooth Overview

The total bandwidth in Bluetooth is divided into 79 channels (each 1 MHz). Frequency hopping (FH) occurs by jumping from one physical channel to another in a pseudorandom sequence. The same hopping sequence referred as an FH channel must be shared by two devices that communicate each other (i.e., by forming a piconet). The hop rate is 1600 hops per second, so that each physical channel is occupied for $625\mu s$. This interval is referred to as a slot and is numbered sequentially. An FH channel is shared between a master and slaves using a time division scheme in which data are transmitted in one direction at a time, with transmissions alternating between two directions. Because more than two devices share the piconet medium (an FH channel), the access technique is Time Division Multiple Access (TDMA). Transmission of a packet starts at the beginning of a slot and can span multiple slots (1, 3, or 5 slots accordingly).The FH sequence is determined pseudorandomly based on piconet master Bluetooth ID. Therefore, several piconets can coexist with minimal interference. Occasionally, two piconets will use the same physical channel during the same time slot, causing a collision and data loss, but this happens rarely, and it is recovered by forward error correction (FEC) and error detection/ARQ techniques. Bluetooth standard defines an asynchronous connectionless (ACL) link for a point-to-multipoint link between the master and all the slaves in the piconet. Achievable data rates on the ACL link vary depending on the number of slots per packet and on the FEC strategy. FEC packets formats are DM1, DM3, and DM5 (with the digits indicating the number of slots used). The non-error coded formats are DH1, DH3, and DH5. Readers can find the details of Bluetooth in the published specification [4] or in the textbook [20]

3 Bluetooth for P2P

A peer in BlueTorrent must be able to discover other peers in order to share content. For two nodes to connect

using Bluetooth, one node should be in the Inquiry state and the other in the Inquiry Scan state. However, since BlueTorrent users are randomly moving, their roles as Inquirers (masters) or Inquirees (slaves) cannot be predetermined, rather, they must be randomly alternated. Bluetooth supports such a random role selection through the periodic inquiry mode. As shown later, the performance of the periodic inquiry mode is dependent on various inquiry parameters. Careful analysis is required to minimize the connection setup latency. In this section, we begin with the overview of the Bluetooth inquiry procedure. We then analyze the mode and empirically find the optimal parameter setting via extensive simulations.

3.1 Bluetooth inquiry procedure

A master peer in the Inquiry state sends inquiry packets and waits for response packets from the potential slave peers. The number of physical channels used for the inquiry procedure is reduced from 79 to 32 for increased discovery efficiency. Moreover, the master peer uses the inquiry hopping sequence for inquiry (as Tx slots), and the potential slave peers uses the inquiry response hopping sequence (different from the former) for response (as Tx slots). Therefore, the master peer must switch to inquiry response hopping sequence to listen to response packets (as Rx slots). Conversely, the potential slave peers must switch to the inquiry hopping sequence to listen to inquiry packets (as Rx slots). The inquiry hopping sequence is divided into two distinct sequences called A- and B-train. Each train contains 16 physical channels and the total duration is 10ms (=16*0.625ms). The Bluetooth specification mandates that each train must be repeated at least 256 times (i.e., 2.56s) before switching to another train. The hopping rate for inquiry is increased to 3200 hops/second (i.e., half-slots) so that the master peer can transmit very short (68μs) inquiry packets every 312.5μs. In each Tx slot, the master can send two inquiry packets, and in next Rx slot, it listens to response packets from other devices. This alternation repeats during the period of the inquiry window $T_{w\_inquiry}$. Note that the host controller interface allows us to set the interval as a *multiple* of 1.28s through $HCI\_Inquiry$ host controller interface (HCI) command [4]. Figure 1 shows an example of the inquiry procedure. The size of the inquiry scan window is 5.12s (=4*1.28s).

Other peers that want to make connections to the master peer should be in the Inquiry Scan state. Each peer enters to the Inquiry scan interval for every inquiry scan interval (T_{inq_scan}) and stays there for inquiry scan window ($T_{w\_inq\_scan}$). $T_{w\_inq\_scan}$ should be greater than the length of a train (i.e., >10ms) in order to ensure that a frequency synchronization between inquiring and scanning peers can happen when the scanning frequency is

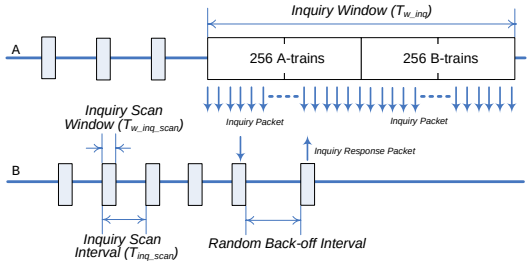


Figure 1. Inquiry procedure example

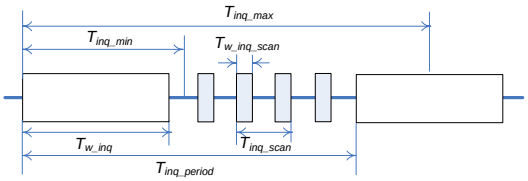


Figure 2. Periodic inquiry mode

in the currently active train of the inquirer, and it cannot be larger than T_{w_inq}. The ranges of T_{inq_scan} and $T_{w\_inq\_scan}$ are given as $[10.625 - 2560ms]$ and $[11.25 - 2560ms]$ respectively. These parameters can be set using $HCI\_Write\_Inquiry\_Scan\_Activity$ HCI command. After receiving the inquiry packet, the peer changes its state to the Inquiry Response state and sends back an inquiry response packet containing its Bluetooth device address and clock information. It is important to note that the peer backs off for a random number of slots to reduce the probability of colliding with other inquiry responses. This random number is drawn uniformly out of a range [0, 1023] (<640ms) if the inquiry scan interval is larger than or equal to 1.28s; otherwise, a range [0, 127] (<80ms) is used. The master peer can receive responses if they arrive within the inquiry window. There is no acknowledgment of a response packet.

3.2 Periodic inquiry mode analysis

The conventional neighbor discovery is asymmetric in the sense that for a given node, the role is fixed to be either as a mater or a slave. In mobile peer-to-peer networks, however, peers should be able to randomly switch their roles in order to find each other, since a node can be either a client or a server. Bluetooth defines a symmetric neighbor discovery mode in the specification (as of v1.0), called $Periodic\_Inquiry\_Mode$ where inquiry procedures are periodically executed. This mode is set by configuring the range of period, $[T_{inq\_min}, T_{inq\_max}]$, and inquiry length ($T_{w\_inq}$) (see Figure 2). In each round, a node alternates an inquiry state immediately followed by an inquiry scan state. The length of an inquiry is fixed to T_{w_inq}, and the length of an inquiry scan state is uniform randomly chosen over $[T_{inq\_min} - T_{w\_inq},\ T_{inq\_max} - T_{w\_inq}]$ in order to avoid synchronization. Note that this interval has nothing to

do with the inquiry scan window ($T_{w\_inq\_scan}$) and interval (T_{inq_scan}).

Although the periodic inquiry mode is widely used, its optimization is not explored. For instance, BlueZ, an official Bluetooth protocol stack for Linux, includes Host Controller Interface Daemon (HCID).[1] HCID provides a wrapper function by internally setting the parameters, i.e., $T_{w\_inq} = 8$ and [$T_{inq\_min} = 16$, $T_{inq\_max} = 24$].[2] Since the unit is 1.28s, each round takes on average 35.84s. Therefore, the parameter selection is not proper for mobile P2P applications.

The goal of this section is to optimize the P2P mode by tuning three key parameters, namely *inquiry window size* (T_{w_inq}), *the length of inquiry scan state* (via [T_{inq_min}, T_{inq_max}]), and *inquiry scan interval* (T_{inq_scan}). First, the inquiry window size may vary based on which Bluetooth version we use. As of Bluetooth v1.2, the interlaced scan mode is used by default. Since the interlaced mode scans two trains in a row, for a given inquiry the probability of missing an inquiry packet (i.e., inquiry failure) is negligible [16]. Bluetooth v1.1, however, does not support it, requiring the inquiry window size T_{w_inq} at least 5 (5*1.28s=6.4s) in order to discover neighbors with high probability [16]. Bluetooth v1.1 devices gradually disappear and currently, v1.2 and v2.0 devices become dominant [12]. Therefore, in this paper we focus only on Bluetooth v1.2 and v2.0.

Second, the length of inquiry scan state is determined by setting [T_{inq_min}, T_{inq_max}]. The minimum value must be carefully chosen such that it should be larger than inquiry scan interval (T_{inq_scan}) since the first scan starts after T_{inq_scan} instead of starting immediately. In addition, the difference between those two values should be greater than zero. Otherwise, the process becomes deterministic – two nodes may never be able to discover each other. It is important to note that although the standard periodic inquiry mode uses the unit of 1.28s for the minimum and maximum values, we can set arbitrary numbers at the granularity of 0.625ms by implementing a periodic inquiry mode in the *application layer*.

Finally, the inquiry scan interval (T_{inq_scan}) is also significant since it determines the usefulness of the inquiry scan state. The usefulness depends on how many "scans" actually happens during the inquiry scan period and can be measured by dividing the average length of inquiry scan period by T_{inq_scan}. The more the number of scans, the higher is the usefulness. As shown later, for a given T_{inq_scan}, there exists an optimal configuration of [T_{inq_min}, T_{inq_max}]. However, this comes at the cost of more energy consumption. This implication is discussed at the end of the section.

Along with the abovementioned parameters, the maximum back-off interval for the inquiry response has a critical impact, especially for Bluetooth v1.2 and v2.0. Even though a "single" inquiry is enough to receive an inquiry packet, an inquiry fails if the node backs off before returning an inquiry packet, and in the mean time the inquiring node switches its role; i.e., the failure probability depends on the maximum back-off interval. Bluetooth specification (v1.2 and v2.0) requires that the back-off interval be drawn uniformly from the range [0,1023] if the inquiry scan interval is larger or equal to 1.28s, or from the range [0,127] otherwise. But the actual implementation is dependent on chipmakers. For instance, in the extended version of this paper we show that Bluetooth v1.2 (Silicon Wave) uses [0,1023], and Bluetooth v2.0 (Broadcom) does not implement random back-off.

3.3 Periodic inquiry mode evaluation

To find an optimal parameter configuration, we developed *InqSim*.[3] This simulator performs a slot-level discrete time, event-driven simulation and accurately models the P2P mode. Every time an event expires, it schedules the next event such as inquiry, scan, and random back-off. During the simulation, actual discovery is checked when a scan expires. A node then examines whether the other party is in the inquiry state and whether the state lasts longer than its random back-off value. If so, the other party will be notified, and finally, it will stop at the end of the current inquiry. To simulate a random encounter of two nodes, the actual measurement starts after passing 160,000 slots (i.e., 100s). Since the overall simulation is simple and requires slot level control and various parameter tuning, we use *InqSim* instead of using UCBT, a simulator used in Section 5.

We measure the discovery latency as the elapsed time until either node successfully discovers the other party. Since we focus on Bluetooth v1.2 and v2.0, inquiry window size 1 is used for all the simulations. For a given range [T_{inq_min}, T_{inq_max}], we define $T_{min} = T_{inq\_min} - T_{w\_inq}$ and $T_{diff} = T_{inq\_max} - T_{inq\_min}$; i.e., the inquiry scan period is in range of [$T_{min}, T_{min} + T_{diff}$]. For given maximum back-off (T_{max_bo}) and inquiry scan intervals, we measure the discovery latency by varying T_{min} and T_{diff}. For each configuration, the average of 5000 runs is presented.

Figure 3 and Figure 4 show the results with the inquiry scan interval of 1.28s, which is the default scan interval. To show the impact of the maximum back-off size, we use $T_{max\_bo} = 1023$ slots (639.375ms) and 127 slots (79.375ms). The figure shows that too small T_{diff} results in high average delay. Given that both nodes are in the

[1] `http://www.bluez.org`

[2] HCID exports functions via D-Bus, a system for *interprocess communication* (IPC)

[3] Available at `http://netlab.cs.ucla.edu/bluetorrent`.

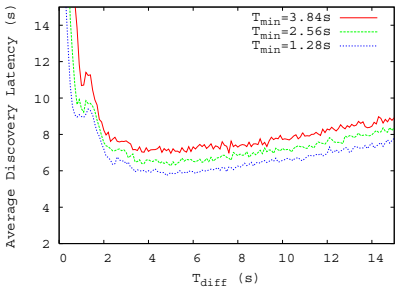


Figure 3. Discovery latency with T_{max_bo}=1023

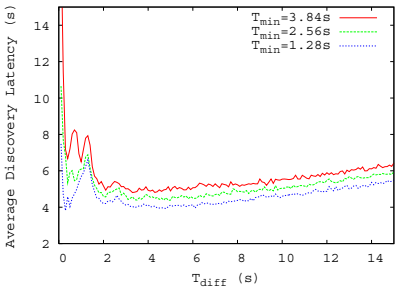


Figure 4. Discovery latency with T_{max_bo}=127

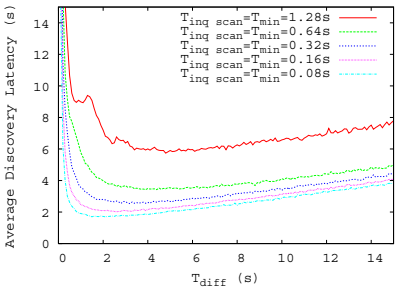


Figure 5. discovery latency with various inquiry scan intervals

same state, they need to spend more time to unlock the sync. As T_{diff} increases, the delay decreases till it reaches a certain threshold. As the inquiry scan period gets longer, randomly encountered nodes are likely in the inquiry scan state. Mathematically speaking, given a collection of random intervals, the length bias or inspection paradox tells us that longer intervals are more likely to be sampled than shorter intervals [17]. Both nodes tend to stay longer in the scan state, thus resulting larger latency. The figure also shows that the back-off value has a great impact on the average latency. When the maximum back-off interval is decreased to 127 slots, the average latency drops more than 30%. Interestingly, Figure 4 shows that the average latency has its minimum at $T_{diff} \approx 0.3s$ and then it increases again till it reaches $T_{diff} \approx 1.5s$. If $T_{diff} < 1.28s$, it does not improve the usefulness of the inquiry scan period; only a single scan per round is feasible. T_{diff} is mainly used to prevent sync. Figure 4 clearly shows that $T_{diff} \approx 0.3s$ is the optimal value, and unnecessarily large T_{diff} adversely affects the performance. In the case of $T_{max\_bo} = 1023$ slots (639.375ms), it requires large enough T_{diff} to also handle random back-off. Let us say that $T_{diff} = 0.1s$. For a given scan period of $[0, 1.38s]$, scan only happens during $[1.28, 1.38s]$. Although an inquiry packet can be successfully received during this period, backing off larger than the residual life time of the inquiry scan period results in failure. From the figures, for 127 slots, $T_{min} = 1.28s$ and $T_{diff} \approx 0.3s$ minimizes the latency to 3.812s. For 1023 slots, $T_{min} = 1.28s$ and $T_{diff} \approx 4.8s$ minimizes the latency to 5.736s. The optimal value can be found when "$T_{inq\_scan} = T_{min}$."

Figure 5 shows the results as a function of inquiry scan interval with $T_{max\_bo} = 1023$. The figure shows the case when $T_{inq\_scan} = T_{min}$, which minimizes the latency. As inquiry scan interval decreases, the average delay decreases. Smaller inquiry scan interval results in more randomness, thus smoothing the curves. The figure also shows that as T_{inq_scan} decreases, the minimum latency tends to converge. For example, for $T_{min} = 0.16s$, $T_{diff} \approx 2.6s$ minimizes the latency to 2.02s; for $T_{min} = 0.08s$, $T_{diff} \approx 2.1s$ minimizes the latency to 1.70s. Therefore, the inquiry scan interval is a *critical* factor for the discovery latency.

In summary, for a given T_{inq_scan}, we show that the optimal T_{min} is equal to T_{inq_scan}, and the optimal T_{diff} can be found through simulations. The results show that optimal values are not multiple of 1.28s, and thus, the standard periodic inquiry mode is suboptimal. We also show that the inquiry scan interval is one of the critical factors in determining the discovery latency. In general, by reducing T_{inq_scan}, we can minimize the discovery latency; e.g., $T_{inq\_scan} = 0.32s$ has 2.53s. One caveat is that this comes at the cost of more energy consumption. In fact, minimizing both discovery latency and energy consumption is two conflicting goals. One of the convincing solutions is to exploit the idea of sociological orbits, a probability mobility model where people move between a set of hubs or information bazaars [10]. Drula et al. propose adaptive energy conserving algorithms based on recent activity level in hubs [8]. For given a set of discovery modes, from aggressive (fast discovery, energy intensive) to lazy (slow discovery, low power consumption), nodes change one's mode based on the contact frequency. P2P content sharing shares the same idea, and thus, for a given set of constraints, our results can be used to optimize the algorithm. Note that readers can find the details of power consumption of each Bluetooth operation in [5].

4 Bluetooth based content sharing protocol

The Bluetooth based P2P connection from the previous section allows random peers to connect to each other. On top of this peer discovery/connection protocol, in this section we present core components of BlueTorrent, namely content sharing and index dissemination modules. Note that in the extended version of this paper [12], we provide a simple mathematical analysis to show the feasibility in a dynamic environment with high churn rate such as in subways or streets.

4.1 Content sharing

BlueTorrent shares contents by using file swarming, mainly due to the limited bandwidth and the short contact duration. The Bluetooth channel tends to be error-prone in the urban streets due to multipath, WiFi interference etc. In addition, the short communication range and mobility of users result in short link/contact duration. For example, two peers moving opposite direction with 1m/s have 10 seconds of link duration. Assuming that peer discovery and connection take 4 seconds, the maximum data size that they can symmetrically transfer is 286KB with DM5 mode. Therefore, it is infeasible for mobile Bluetooth nodes to share a relatively large file without using file swarming.

In BlueTorrent, the data source (i.e., static BT-AP) divides a file into K pieces or blocks. BlueTorrent peers can download pieces from static BT-APs and other peers. Each node has a bitmap of the available pieces for efficient piece reconciliation. Whenever a connection is available, peers first exchange their bitmaps to find out missing pieces through simple bit operations. The size of a typical bitmap is as small as tens of bytes even if there are more than 100 blocks. To be more precise, given K pieces, the size of a bitmap table is $\lceil K/8 \rceil$ bytes. For example, the size of a bitmap table for 100 blocks takes 13 bytes. Given 286KB of average data transfer per contact with DM5 mode, the overhead is negligible. It is interesting to note that when only one party has data to transfer, it uses asymmetric mode whereas a symmetric mode is used when both have data to transfer.

The size of a piece should be carefully selected based on the characteristics of Bluetooth bandwidth and mobility patterns. If the block is too big, peers cannot download a single block during their contact duration. A block can be divided into very small sub-blocks, say, with size 1KB. However, this costs additional bandwidth and computation overhead for sub-block level reconciliation. Through a simple mathematical model we show that in the dynamic environment with limited bandwidth, sharing a large file, i.e., a large number of blocks, leads to ineffective file swarming [12].

4.2 Index dissemination

The prerequisite of content distribution is to know where the content is. Content can be searchable through indices which include a unique ID (e.g., 32 bit hash value of the content), title, producer, media type, etc. A static BT AP is a publisher of content, and it pushes index to mobile nodes. Users who are interested in specific information can proactively query other peers. A user can express his request for contents as a simple query string. For example, those who are interested in downloading a movie preview clip of "Pirates of the Caribbean" will prepare a query string such as "Pirates & Caribbean" with media type "avi/mpg." We assume that BlueTorrent is equipped with a lightweight database for index management and search. Since users have a limited, often local scope of files of interest, the size of the indices is small. Thus, we can assume that the storage overhead of index dissemination is minimal compared to the actual content size. Whenever a node discovers another node, it first sends the query. Upon receiving a query, the target node will look up its database to find an exact match of keys. The node could employ sophisticated similarity measures such as [11]. The meta-data match will be automatically reported to the query originator, and the user may decide whether to initiate downloading using the ID of the content.

Moving Area (Xr, Yr)	[25, 50, 100] $\times$ [3, 5] m^2
Number of Nodes($\|N\|$)	25, 50, 75, 100
Number of AP Nodes (N_{AP})	2
Moving speed of nodes (S)	[0.0, 0.4, 0.8, 1.2, 1.6] m/s
Packet type (P)	DH5
Block Size/Number (B_S, B_N)	[(6KB, 200), (12KB, 100), (24KB, 50)]
Transfer time (T_t)	10 sec

Table 1. Simulation Parameters

5 Simulation

In this section, we evaluate the performance of BlueTorrent using UCBT NS-2 extension, a Bluetooth simulator that is publicly available and open source.[4] UCBT implements the majority of the protocols in the Bluetooth including baseband, LMP, L2CAP, and BNEP.

5.1 Simulation Setup

Mobility – We assume Bluetooth device users are in a corridor that has a fixed boundary. Users are moving with specific waypoint as from East to West or from West to East. Waypoints are randomly chosen in the initial stage. Maximum speed (0.0, 0.4, 0.8, and 1.6 m/s) is predefined to limit node's speed. 1.6 m/s is selected because this speed is 5.76 km/h and it is about 1.5 times faster than regular walking speed. To add a random factor, direction is changed periodically with an offset in the range [-10, 10] degrees with respect to the original direction. When a node reaches North or South bound of a simulation area, it is mirrored back in the simulation area. When a node reaches East or West bound, we regard that the node moves out of corridor. So, we eliminate that node and regenerate a new node at the other bound (for example, if a node went out of the East bound, a new node starts at the West bound). The regenerated node has same speed and direction of the eliminated

[4] UCBT: `https://www.ececs.uc.edu/cdmc/ucbt`
NS-2: `http://www.isi.edu/nsnam/ns/`

node.

When a node is eliminated and a new node regenerated to substitute the eliminated node, we choose *reset* or *no-reset* mode. The reset mode deletes all stored data in the eliminated node and the regenerated node starts with empty data. So, it simulates the situation where one node is going out of an area and a new node is coming into the area. The no-reset mode, on the other hand, transfers all stored data from the eliminated node to the regenerated node. So, it is as if the node going out of one side re-enters from the other side.

Test Scenarios – There are two types of nodes: static APs and mobile nodes. APs have all data set and transfer data to mobile nodes. They are randomly located in the area. Mobile nodes start without data, and they pull data from APs or other mobile nodes. In the *AP mode*, only the AP does inquiry and connects to mobile nodes and transfers data. In the *P2P mode*, every node (AP or mobile node) can connect to the other nodes and exchange data. Initially, only APs can transfer data. Received data blocks from APs are then re-distributed in a P2P fashion. In the simulations, APs distribute a single file. Since the transfer of meta-data is negligible compared to data transfer, we assume that each node already has meta-data of a file.

Metrics – We measure the *download finish time* for *no-reset* mode. In the case of *reset* mode, nodes usually are not able to download a file completely, instead, we measure the progress using *download percentage* of all the nodes that have passed the simulated area during the simulation. By dividing the summation of the downloaded blocks by the nodes, we can calculate the expected number of downloaded blocks. As time tends to infinity, by the strong law of large number, this is equal to the ensemble average of the number of blocks that a random node can download while passing by the simulated region.

Inquiry and Transfer Stages – In the Inquiry stage, nodes perform inquiry and inquiry scan periodically (i.e., P2P mode). When a node discovers others, it chooses one of the nodes as a master. Peer selection could be a potential problem. To deal with this issue, each node keeps a connection log, which contains a list of peers with discovered time and last connection time. If there are nodes without any connection, the node randomly chooses one. Otherwise, it chooses the earliest connected node, which potentially has met others and carries more fresh blocks than others. This selection is automatically logged. Transfer stage begins after creating a connection to the selected node. Nodes first exchanges block bitmaps, which contain flag bits of data block, to identify useful blocks. If there are useful blocks, data blocks are transferred. Otherwise, the connection is terminated. The master node then selects another node and restarts the overall procedure. If there no more node to connect to, it goes back to the Inquiry stage.

Parameter Summary – We choose as moving area a corridor model that has a relatively long length compared to width. Unless otherwise mentioned, an area of $100\times5\ m^2$ is used by default. The number of nodes varies to change density while the number of AP is fixed to two. Nodes move at the speed up to 1.6 m/s. DH5 packet type is used because this has the highest data rate among all ACL packet types in Bluetooth v1.2. We use 1.2MB size file to transmit and this file is divided as 200, 100, or 50 blocks. Transfer time is set as 10 seconds to tradeoff between reducing inquiry overhead and reducing out-of-range possibility during communication. When transfer time is long, inquiry overhead is reduced but nodes can be exit the communication range more frequently during transmission. When transfer time is short, inquiry overhead is increased. Details of parameters are shown in Table 1.

5.2 Simulation Results

In this section, we mainly compare the performance of P2P and AP modes. The overall download progress is first presented to show the benefits of the P2P mode. We then show the impacts of various parameters, namely the average speed of mobile nodes, the number of blocks, and the corridor length.

Overall download progress – Figure 6 and Figure 7 show average download percentages of both *reset* and *no-reset* modes for every 10 seconds. Total 50 nodes are moving in an area of $100\times5\ m^2$, and one hundred blocks are used. The figure shows that the P2P mode has three times better performance than the AP mode. Multiple peer-to-peer connections increase transmission chances and throughput. In the case of *no-reset*, although at the beginning nodes collect blocks almost linearly over time, after passing around 50%, we see that it takes progressively longer to collect new blocks. This problem is known as a coupon collection problem: the more the coupons you collect, the higher is the probability of collecting an overlapping coupon. This problem can be mitigated by using coding techniques, namely source or network coding [7][15]. Evaluating these techniques is part of our future work. In [14], readers can find some preliminarily results of using network coding in mobile ad hoc networks.

Impact of speed – Figure 8 shows average download finish time of nodes. $100\times5\ m^2$ moving area and 100 Blocks are used. The figure shows that as speed increases the download finish time increases. When nodes are static (i.e., speed = 0.0m/s), the density of nodes is critical; if it is below a certain threshold, some nodes may not be able to download any blocks. In the case of the P2P mode, the average finish time is not sensitive to density of nodes. Interestingly, the AP mode has an optimal speed that minimizes the download finish time. Two APs are shared among all the nodes.

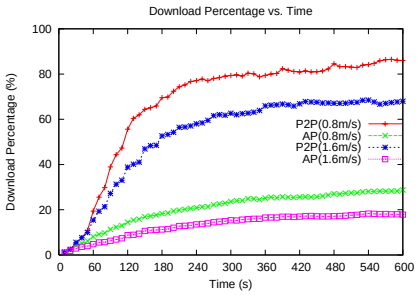


Figure 6. Download Percentage (reset)

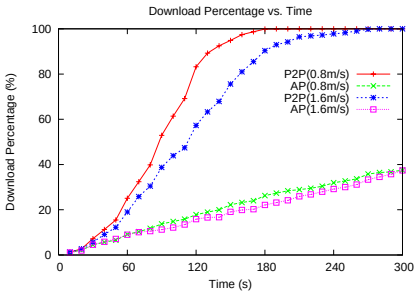


Figure 7. Download Percentage (no-reset)

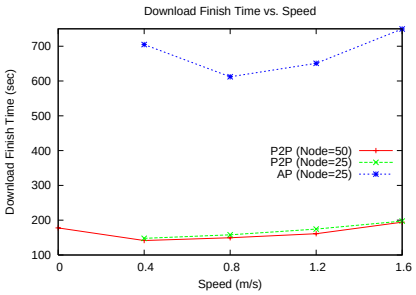


Figure 8. Finish Time vs. Speed

If nodes move slowly, it is possible that at some point AP is idle (owing to low density, N=25). The idle period is mainly dependent on the speed; as speed increases, the idle period decreases. However, if a node moves too fast, the effectiveness of a connection, which is calculated by dividing the data transfer time by the total connection duration (i.e., discovery+data transfer), decreases. Thus, fast mobility adversely affects the performance. The figures shows that when nodes move at the speed of 0.8m/s on average, the AP mode has the best performance.

Figure 9 shows average numbers of helpful connections and unhelpful connections of nodes. 25 nodes are moving in an area of 100×5 m^2 and share a 100 block file. The number of unhelpful connections for P2P mode significantly decreases, as speed increases. Fast mobility blends nodes well, thus reducing the chances of having unhelpful connections. The AP mode also experiences a slight decrement of the number of unhelpful connections, since the total number of connections is much smaller than that of P2P case. On the other hand, the number of helpful connection increases up to a certain threshold (P2P case $0.8m/s$, AP case $0.4m/s$). Thus, we conclude that the gain of fast mobility comes from the relative increment of helpful connections.

Impact of the number of blocks – Figure 10 shows average download percentage of nodes at the 200 second mark as a function of speeds with different number of blocks. In general the download percentage is not sensitive to the number of blocks. Although the overhead of L2CAP level acks increases as the number of blocks increases, its impact is not significant. Instead, mobility has a greater impact. With fast mobility a node will experience more frequent disconnections, causing cancelation of the currently downloaded block. If the size of a block is large, this will incur non-negligible performance degradation. Thus, when the speed is 1.6m/s, the largest block size (BL=200) performs the best.

Impact of the corridor length – Figure 11 shows download percentage of nodes at the 200 second mark with different corridor length. When nodes are static (0.0 m/s), there is no mobility; thus, node reset does not happen. So, length of corridor only affects density of nodes and short corridor length shows better performance. When nodes are mobile (0.4-1.2 m/s), short corridor length makes more frequent node reset than longer ones therefore decreases download percentage. Note that density increment in the P2P mode has much greater impact on the performance. As explained before, the performance of the AP mode is not significantly improved by the density after a certain threshold.

6 Experiments

In this section, we show our preliminary testbed implementation results. In the experiment, we use BlueZ Bluetooth protocol stack for Linux [5] and Bluetake BT009Si (Silicon Wave, Bluetooth v1.2). BlueZ consists of HCI Core, HCI USB device driver, L2CAP protocol module, and testing utilities. We used 3 desktops and 5 laptops which have RedHat 9.0 with similar configurations of Pentium IV with 512MB RAM. We place two nodes around the AP, and the rest of the nodes are located about 10m away from the AP.

To emulate mobility we use the following method. The AP is up for a certain percentage of a given period, and for the rest of the period it is down. This emulates nodes moving out of the AP's communication range. During this period (i.e., when AP is down), only P2P mode can transfer data. Similar to the simulation section, we use the lifetime of the nodes that are moving at maximum speeds of 0.8m/s and 1.6m/s in a 100m length corridor. Nodes are reset at the end of their lifetime to emulate the maximum moving speeds.

Figure 12 shows average download percentage of nodes every 10 seconds. For Figure 12(a), nodes are reset according to the lifetime model. But, for Figure 12(b), they are not reset. P2P mode shows about twice better performance than AP mode because multiple simultaneous peer-to-peer connections increase transmission chances and throughput. In Figure 12(a), the download percentages of both mode increase fast at the beginning, because all nodes have no data. After a while, AP case download percentage drops because AP is down (i.e., nodes are out of AP's communica-

[5] `http://www.bluez.org`

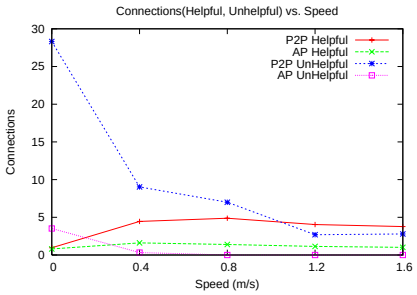


Figure 9. Connection status

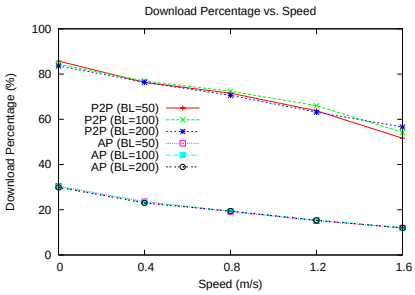


Figure 10. Number of blocks

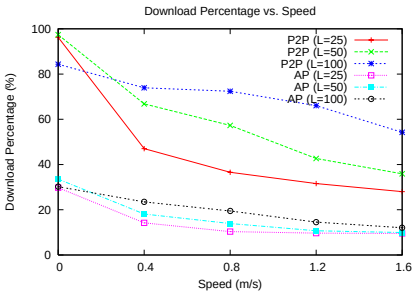
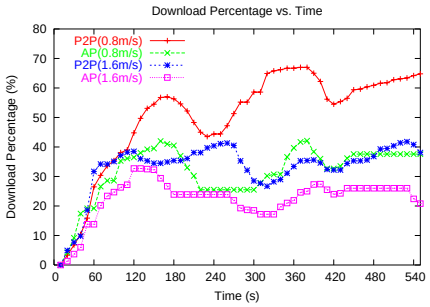
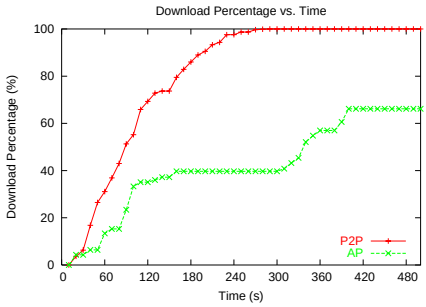


Figure 11. Corridor length



(a) reset



(b) no-reset

Figure 12. Download Percentage (Testbed)

tion range). However, P2P case download percentage continues to increase because of simultaneous transmissions among nodes. In the figure, the AP down period is shown by non-changing download percentage of AP mode. When reset happens, new nodes are generated and thus the average download percentage drops for both AP and P2P cases. In AP mode, this drop shows steeper slope than in P2P case if AP is down during this period. Reset does not happen in Figure 12(b) and thus decrement of download percentage does not happen. However, we still have AP down period in which AP mode shows no change in download percentage during this period. P2P case continuously increases and the average download percentage reaches 100% far earlier than AP mode. AP mode does not reach 100% average download percentage during the test time.

7 Related work

The P2P mode introduced in our paper is also known as symmetric discovery mode since each device alternates their roles as master and slave. To this end, $Periodic\_Inquiry\_Mode$ HCI command was introduced as of Bluetooth specification v1.0. Inquiry window size (T_{w_inq}) is fixed and the inquiry period (i.e., the interval in between two consecutive inquiries) is chosen uniform randomly over $[T_{inq\_min},\ T_{inq\_max}]$. To find average discovery latency, Salonidis et al. [18] simplified the the model, making it analytically tractable: both inquiry and inquiry scan interval are treated random, and inquiry scan interval (T_{inq_scan}) is not considered. But the model hardly reflects a real Bluetooth device, and it cannot be used to optimize the periodic inquiry mode. In fact, it is non-trivial to analytically model the system. In this paper, we accurately simulate the mode and find the optimal parameter configuration. Moreover, we show that the inquiry scan interval, which was neglected in [18], has a great impact on the average latency. Instead of optimizing the periodic inquiry mode, Siegemund et al. [19] proposed a cooperative peer discover protocol. A few nodes per piconet perform cooperative discovery; i.e., each node inquires others and then, proactively exchanges the results. Our finding can be applied to this scheme to further expedites the discovery.

Bohman et al. [3] proposed a variant of the periodic inquiry mode: for each round, the duration of inquiry and inquiry scan states is randomly selected over $[min,\ max]$. We suspect that they authors mistakenly proposed to choose the inquiry duration at random: the length of inquiry state should be multiple of 1.28s since the unit of inquiry window is 1.28s. The problem persists in [8], since they adapted this method. In their scheme, the duration of inquiry and inquiry scan states is chosen over $[C_{inq},\ C_{inq} + 2V_{inq}]$ and $[C_{scan},\ C_{scan} + 2V_{scan}]$ respectively, where $C_{inq/scan}$ represents the fixed part and $V_{inq/scan}$ the variable part. We assume that these variants were proposed because the authors were unaware of the periodic inquiry mode in the specification.

Aalto et al. introduces a B-MAD system for delivering permission-based “location-aware” mobile advertisements

to mobile phones using Bluetooth positioning and Wireless Application Protocol (WAP) Push [1]. They installed nine Bluetooth sensors (Nokia 3650 phones running the Bluetooth Sensor software) in the display windows of eight retail stores. The store produced eleven advertisements containing some special offers and discounts. They measured positioning time (time to discover a new device), positioning accuracy, scalability, and latency.

BlueCasting is a widely accepted proximity marketing system.[6] BlueCasting servers can be deployed at poster sites, retails, etc. such that it can identify Bluetooth users and deliver tailored messages. The advertisement log is managed in a central database in order to prevent receiving redundant advertisements. BlueBlitz Magic Beamer supports similar functionalities.[7] In particular, it provides two way messaging/advertisements and the Internet access as a hotspot. Although these devices support class 1, i.e., 100m communication range, typical Bluetooth devices only support class 2, i.e., 10 m, and thus, mobility of users has a huge impact on the performance. As mentioned before, with an error-prone channel due to multipath, WiFi interference etc., downloading bandwidth is limited; downloading several mega byte files without stopping near the APs is not feasible. BlueTorrent remedies this problem by letting mobile users cooperatively carry and forward data to minimize downloading delay.

A Bluetooth Content Distribution (BCD) station supports content distribution in a "static" environment such as a bus [13]. A BCD station is also equipped with WiFi, and with help of opportunistic connections, it can be synchronized. Once synchronized, data will be distributed to the Bluetooth end users while they are on board. Like other previous products, BCD does neither consider P2P-based content distribution, nor mobility of users.

8 Conclusion

In this paper we proposed BlueTorrent, a cooperative content sharing for mobile Bluetooth users. We analyzed and found the optimum setting for the periodic inquiry mode in order to minimize inquiry/connection time. We then proposed and analyzed index dissemination and file swarming protocols in dynamic, sparse networks. Simulation results showed that cooperative P2P file sharing achieves a greater performance improvement in download time compared to the traditional AP only mode. Finally, via testbed experiments we showed the feasibility of BlueTorrent.

[6] BlueCasting: `http://www.bluecasting.com`
[7] BlueBlitz: `http://www.blueblitz.com`

References

[1] L. Aalto, N. Gothlin, J. Korhonen, , and T. Ojala. Bluetooth and WAP Push Based Location-Aware Mobile Advertising System. In *MobiSys'04*, Boston, NY, Jun. 2004.

[2] BBC NEWS: Music trial taps into Bluetooth. `http://news.bbc.co.uk/1/hi/technology/4392534.stm`.

[3] D. Bohman, M. Frank, P. Martini, and C. Scholz. Performance of Symmetric Neighbor Discovery in Bluetooth Ad Hoc Networks. In *German Workshop on Mobile Ad-hoc Networking (WMAN'04)*, Ulm, Germany, Dec. 2004.

[4] Bluetooth SIG. Bluetooth Specification v2.0, 2004.

[5] J.-C. Cano, J.-M. Cano, E. Gonzalez, C. Calafate, and P. Manzoni. Power Characterization of a Bluetooth-based Wireless Node for Ubiquitous Computing. In *ICWMC'06*, Bucharest, Romania, July 2006.

[6] CBS Goes Bluetooth To Promote Fall TV Line-Up. `http://www.telecomweb.com/tnd/18847.html`.

[7] P. Chou, Y. Wu, and K. Jain. Practical Network Coding. In *51st Allerton Conf. Communication, Control and Computing*, Allerton, Oct. 2003.

[8] C. Drulă. Fast and Energy Efficient Neighbour Discovery for Opportunistic Networking with Bluetooth. *Master Thesis, Dept. of Computer Science, University of Toronto*, 2005.

[9] The Economist: Bluetooth's quiet success. `http://www.economist.com/science/tq/displayStory.cfm?story_id=7001843`.

[10] J. Ghosh, S. Yoon, H. Q. Ngo, and C. Qiao. Sociological Orbit for Efficient Routing in Intermittently Connected Mobile Ad Hoc Networks. In *Technical Report TR-2005-19*, University at Buffalo, Apr. 2005.

[11] T. Hofmann. Probabilistic Latent Semantic Analysis. In *UAI'99*, 1999.

[12] S. Jung, U. Lee, A. Chang, D. Cho, and M. Gerla. BlueTorrent: Cooperative Content Sharing for Bluetooth Users. Technical report, UCLA, Dec. 2006.

[13] J. LeBrun and C.-N. Chuah. Feasibility Study of Bluetooth-Based Content Distribution Stations on Public Transit Systems. In *ACM MobiShare'06*, Los Angeles, CA, Sept. 2006.

[14] U. Lee, J.-S. Park, J. Yeh, G. Pau, and M. Gerla. CodeTorrent: Content Distribution using Network Coding in VANETs. In *MobiShare'06*, Los Angeles, CA, Sept. 2006.

[15] P. Maymounkov and D. Mazieres. Rateless Codes and Big Downloads. In *IPTPS'03*, Berkeley, CA, Feb. 2003.

[16] B. S. Peterson, R. O. Baldwin, and J. P. Kharoufeh. Bluetooth Inquiry Time Characterization and Selection. *IEEE Transactions on Mobile Computing*, 5(9):1173–1187, Sep. 2006.

[17] S. M. Ross. *Introduction to Probability Models*. Academic Press, 2002.

[18] T. Salonidis, P. Bhagwat, L. Tassiulas, and R. O. LaMaire. Distributed Topology Construction of Bluetooth Personal Area Networks. In *INFOCOM*, Anchorage, AK, Apr. 2001.

[19] F. Siegemund and M. Rohs. Rendezvous Layer Protocols for Bluetooth-Enabled Smart Devices. *Personal and Ubiquitous Computing Journal*, 7(2):91–101, Jul. 2003.

[20] W. Stallings. *Wireless Communications and Networks*. Prentice Hall, 2005.