P2P Content Distribution to Mobile Bluetooth Users

Uichin Lee, *Member, IEEE*, Sewook Jung, Dae-Ki Cho, Alexander Chang, Junho Choi, and Mario Gerla, *Fellow, IEEE*

***Abstract*—The use of handheld devices, such as smart phones for personal entertainment, has become commonplace in today's lifestyle. Virtually all of these devices are equipped with Bluetooth technology, which can be used to distribute entertainment content, such as music and movie clips. Mobile users can download content from opportunistically available infrastructure (e.g., digital billboards) and direct peer-to-peer (P2P) collaboration, which significantly increases content availability/coverage. P2P content distribution protocol design is heavily influenced by the characteristics of Bluetooth, which is a main departure from Internet-based content distribution. However, little has been done to understand the performance of overall Bluetooth operations, ranging from peer discovery to data downloading, in *dynamic* environments with mobility, interference, and different Bluetooth versions/chipsets. In this paper, we perform an extensive measurement study and find that Bluetooth-based content distribution suffers from time/energy-consuming resource discovery and limited bandwidth, even with the enhanced features of the latest Bluetooth version. Given this, we discuss strategies that can effectively improve the performance of the resource-discovery and downloading phases.**

***Index Terms*—Bluetooth, content distribution, measurement.**

I. Introduction

SMALL portable devices equipped with one or more wireless technologies (e.g., WiFi, Bluetooth, and ZigBee), ranging from smart phones (e.g., iPhone and Blackberry) to digital music players (e.g., Microsoft Zune and iPod Touch) are becoming increasingly popular. Beyond the means of wireless synchronization as cable replacement, such radio technologies are paving the way for novel applications to mobile users, thus delivering new opportunities to all facets of the industry. In particular, this paper deals with one fast-growing application, i.e., proximity advertising and marketing using Bluetooth-enabled digital billboards such as BlueCasting [1] and BlueBlitz Magic Beamer [2].

So far, mobile users have directly been tapping this abundance of data from the opportunistically available digital billboards. Given the fact that such billboards are located at popular places (e.g., subway stations, department stores, tour sites, and stadiums) and more users simultaneously download from the network while on the move, it is natural to expect that several of them may be interested in downloading the same content. Thus, it makes sense to explore the extension of *opportunistic* networking to include direct peer-to-peer (P2P) exchanges. There are obvious advantages to this P2P and infrastructure downloading synergy. P2P technology enables us to overcome the short contact duration with a Bluetooth access point (BT-AP) due to the short communication range (10 m) and mobility, obviating the need to install BT-APs every 10 m. Thus, our goal is to devise an efficient P2P content distribution protocol in a mobile environment with opportunistic infrastructure supports.

A good model is offered by Internet-based P2P indexing (resource discovery) and content distribution protocols. A logical overlay network (e.g., Gnutella) is built on top of the Internet to provide efficient ways of resource discovery, even with the dynamics of user participation (called churning). In addition, BitTorrent-like P2P file swarming is commonly used for content distribution, where content is divided into a number of small pieces, and users cooperatively share whatever pieces they have [3]. Then, the question is whether we can adapt these techniques to mobile environments. The overlay approach may not be feasible in mobile environments, because we cannot assume *Internet-like connectivity* [4], [5]. Data-access patterns become opportunistic since data can be exchanged whenever there *is* a user of the same interests or infrastructure in the neighborhood. BitTorrent-like P2P file swarming should be used to efficiently exploit the opportunistic connectivity. Thus, our focus is on *opportunistic P2P file swarming* in mobile environments.

P2P protocol design in mobile environments is dependent on the underlying radio technology, which is a main departure from that in the Internet. P2P file swarming is generally divided into two phases, i.e., resource discovery (in the immediate neighborhood) and downloading. The protocol design of each component must carefully consider the characteristics of Bluetooth to obtain better performance. However, little efforts have been made to understand the performance of overall Bluetooth operations, ranging from peer discovery to data downloading, in dynamic environments with mobility and interference (e.g., WiFi and obstacles). In addition, the performance variation among different Bluetooth versions was not examined, although different Bluetooth versions are currently populated in the

Manuscript received February 25, 2009; revised June 8, 2009. First published August 25, 2009; current version published January 20, 2010. This work was supported in part, through participation in the International Technology Alliance, by the U.S. Army Research Laboratory and the U.K. Ministry of Defense under Agreement Number W911NF-06-3-0001, and in part by ARMY MURI under Funding W911NF0510246. The work was done while U. Lee, S. Jung, A. Chang, and J. Choi were affiliated with University of California at Los Angeles. The review of this paper was coordinated by Prof. Y.-B. Lin.

U. Lee is with Bell Laboratories, Alcatel-Lucent, Holmdel, NJ 07733 USA (e-mail: uclee@ieee.org).

S. Jung is with Broadcom Cooperation, San Diego, CA 92127 USA (e-mail: sewookj@cs.ucla.edu).

D.-K. Cho and M. Gerla are with the Department of Computer Science, University of California at Los Angeles, Los Angeles, CA 90095 USA (e-mail: pinecho@cs.ucla.edu; gerla@cs.ucla.edu).

A. Chang is with Yahoo!, Inc., Burbank, CA 91504 USA (e-mail: acmchang@cs.ucla.edu).

J. Choi is with Heavy Iron Studios, Los Angeles, CA 90045 USA (e-mail: junho.choi@heavy-iron.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVT.2009.2030893

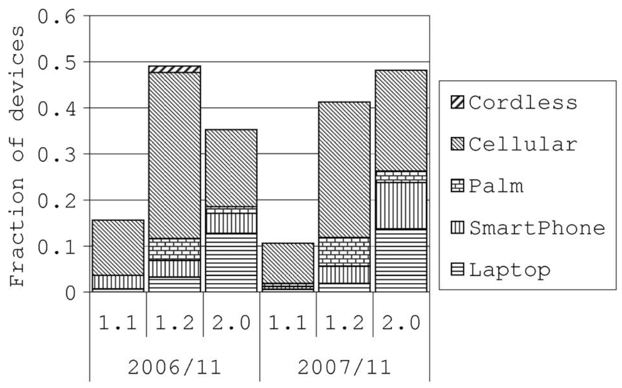


Fig. 1. Bluetooth version distribution. The statistics were gathered at the University of California at Los Angeles (UCLA) campus on November 29–30, 2006, and November 29, 2007. Bluetooth version information was collected from those devices that allowed making connections (75% of 402 devices in 2006 and 60% of 351 devices in 2007).

market due to the slow adaptation of Bluetooth standards, as shown in Fig. 1. Thus far, measurement studies have reported only the performance of specific operations in static scenarios with a certain Bluetooth version [6]–[9]. For instance, Kasten and Langheinrich [7] reported the issues of power consumption and peer discovery latency in a customized Bluetooth-based sensor node, and Leopold showed the peer discovery and throughput measurement results of a Bluetooth v1.1 device [8]. As a result, the impacts of the Bluetooth characteristics on protocol design and evaluation were not addressed in recent content distribution systems [10]–[14].

In this paper, we perform extensive experiments to accurately characterize the Bluetooth performance in a dynamic mobile environment. We emulate a mobile user with an Amigobot, which is a programmable robot that can travel at a speed of up to 2.2 m/s [15]. Our measurements include peer discovery, connection setup, and download bandwidth under various conditions, such as mobility, interference, and different Bluetooth versions. To the best of our knowledge, this is the first experiment with Bluetooth in an externally controlled mobile environment. The following summarizes our main findings.

1) Resource discovery in Bluetooth is time/energy consuming, even with the latest Bluetooth version 2.0. In Bluetooth, one must first discover its neighbors (peer discovery) and then sequentially create a connection to each neighbor to resolve one's query (content discovery). In Section IV, for efficient resource discovery, we present a cooperative resource discovery method that reduces resource discovery latency, and an energy efficient peer discovery protocol that saves energy with minimal performance penalty.
2) Mobile Bluetooth users experience significant throughput drop. In particular, the measured throughput widely varies over distance and is much lower than the simulation results. We discover that this is mainly due to the autorate control algorithm in Bluetooth, which is called the channel-quality-driven data rate (CQDDR) implementation in Bluetooth devices. Since the autorate control algorithm does not reveal its current packet type information to the upper layer (i.e., Bluetooth software stack such as BlueZ), the incorrect choice of a packet type at the upper layer results in a significant performance loss. In Section IV, we remedy this problem using a receiver feedback scheme that enables dynamic packet size selection at the Bluetooth software stack level.
3) We find that Bluetooth devices are not optimized for "mobile" environments. Advanced features such as transmission power control, which adjusts power to save energy, and adaptive frequency hopping (AFH), which prevents the use of interfered channels, do not work well, mainly because of their long response time, as opposed to a short contact period in mobile scenarios.
4) We report that the type of Bluetooth versions/chipset has a great impact on peer discovery, connection setup, and data throughput.

Note that our measurement results can also be applicable to optimizing networking protocols. For instance, delay-tolerant network routing [17], [18], which aims at providing reliable data delivery, even with connectivity disruptions (due to mobility and sparse node density), can maximize data transfer per contact, thereby increasing the end-to-end throughput. The rest of this paper is organized as follows: Section II introduces the basics of Bluetooth. Section III describes the experimental setup and presents our measurement results. Section IV discusses various schemes to enhance the performance of a file-swarming protocol in Bluetooth. Related work is reviewed in Section V. Finally, Section VI concludes this paper.

II. Background

Bluetooth is defined as a layered-protocol architecture. Radio specifies details of the physical layer of Bluetooth. Baseband concerns peer discovery, connection setup, addressing, packet format, power control, timing, etc. The Link Manager Protocol (LMP) is responsible for link setup between Bluetooth devices and link management. This includes the control and negotiation of baseband packet size and transmission power. The logical link control and adaptation protocol (L2CAP) adapts upper layer protocols (e.g., segmentation and reassembly, protocol multiplexing, flow control per L2CAP channel, error control, and retransmissions) to the baseband layer via the host control interface (HCI). In this section, we review the radio and baseband of Bluetooth. Details on the Bluetooth radio system can be found in [19] and [20].

In Bluetooth, the total bandwidth is divided into 79 channels (with each having 1 MHz). Frequency hopping (FH) occurs by jumping from one physical channel to another in a pseudorandom sequence. The hop rate is 1600 hop/s, so that each physical channel is occupied for 625 μs. This interval is referred to as a slot and is sequentially numbered. The basic unit of networking in Bluetooth is a *piconet*, which consists of a master and one to seven active slave devices. The master determines a hopping sequence based on its Bluetooth ID (which is referred to as an FH channel), which shall be used by all the devices on this piconet. The FH channel is shared between a master and slaves using time-division duplex, i.e., data are transmitted one direction at a time, with transmission alternating both directions. Multiple slaves share the piconet medium using time-division multiple

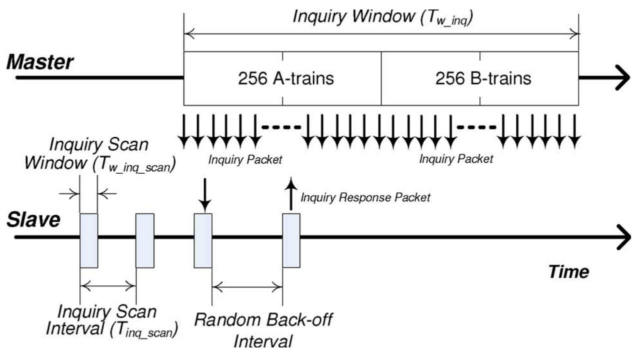


Fig. 2. Bluetooth inquiry procedure example. The size of the inquiry window is 5.12 s $(= 4 \times 1.28$ s). The master node sends inquiry packets during the inquiry window. The slave node periodically listens to a specific channel to wait for an inquiry packet (inquiry scan). After receiving an inquiry packet, the slave performs a random backoff and then sends an inquiry response.

access. This FH channel is a form of code-division multiple access; thus, several piconets can coexist with minimal interference. Two piconets will occasionally use the same physical channel during the same time slot, causing a collision and data loss, but this rarely happens, and it is recovered by forward error correction (FEC) and error detection/automatic repeat request (ARQ) techniques. Multiple piconets in the same area are referred to as a *scatternet*, and different piconets can be interconnected. However, scatternet connection support is defined as optional in all Bluetooth specifications; thus, many Bluetooth chips do not support scatternets [13].

Physical Link and Packet Types: The Bluetooth link supports synchronous services such as voice traffic via the synchronous-connection-oriented link and asynchronous services such as bursty data traffic via the asynchronous connectionless (ACL) link. Since we focus on the data traffic, only the ACL link is illustrated in the following. Baseband defines an ACL link to provide packet-switched connection between the master and active slaves (with one ACL link per slave). The master interleaves traffic between multiple active slaves (up to seven slaves).

Achievable data rates on the ACL link vary, depending on the number of slots per packet (one, three, and five slots) and on the FEC strategy. The more the number of slots, the larger the payload size. The FEC packet formats are DM1 (17 B), DM3 (121 B), and DM5 (224 B), and the nonerror-coded formats are DH1 (27 B), DH3 (183 B), and DH5 (339 B). For instance, DH5 can achieve the maximum rate of 732.2 and 433.9 kb/s for asymmetric and symmetric communications, respectively. The maximum asymmetric and symmetric data rates are 732.2 and 433.9 kb/s with DH5, respectively. As for the Bluetooth v2.0 enhanced data rate (EDR), phase-shift keying (PSK) modulation has been added. PSK increases the coding bits per symbol to 2 and 3 bits, thus introducing EDR packet types *without FEC* as 2-DH1/2-DH3/2-DH5 (packet size $=$ $2 \times \text{DH}x$ and maximum asymmetric and symmetric rate $=$ 1448.5 and 869.7 kb/s, respectively) and 3-DH1/3-DH3/3-DH5 (packet size $== 3 \times \text{DH}x$ and maximum asymmetric and symmetric rate $= 2178.1$ and 1306.9 kb/s, respectively).

Peer Discovery: The first step in establishing a piconet is to identify devices in range that wish to participate in the piconet. As shown in Fig. 2, the inquiry procedure begins when the potential master transmits an ID packet with *Inquiry Hopping Sequence* (Tx slot) and waits for the response packets from the slaves with the *Inquiry Response Hopping Sequence* (Rx slot). The Inquiry Hopping Sequence consists of 32 unique wake-up frequencies equally distributed over the 79 frequencies and is divided into two distinct sequences called A and B trains. Each train contains 16 physical channels and must be repeated at least 256 times (i.e., $2.56 \text{ s} = 256 \times 10$ ms) before switching to another train. During the inquiry window $(T_{\text{w\_inq}})$, the master repeats the following: Send two inquiry packets in each Tx slot, and then, listen to response packets from other devices in the following Rx slot. The potential slaves periodically (i.e., $T_{\text{inq\_scan}}$) enter the Inquiry scan state and listen to a specific channel to search for inquiry packets for the inquiry scan window $(T_{\text{w\_inq\_scan}})$. When a node receives an inquiry packet, it enters the Inquiry Response state and returns a frequency hop synchronization (FHS) packet (i.e., inquiry response) after backing off a random number of slots to avoid collision. An FHS packet contains the device address, page scan repetition mode (PSRM), and clock offset, which are required by the master to initiate a connection. As of Bluetooth v1.2, the *interlaced inquiry scan* is introduced. In the original scheme, because the master repeats a specific packet train many times, a slave node could miss the whole packet train if it listens to a channel that belongs to the other packet train (e.g., inquiry A train/scan B train or inquiry B train/scan A train). To prevent such pathological situations, the interlaced scan allows a slave node to scan two trains (i.e., A and B) in a row. Thus, the probability of missing an inquiry packet becomes negligible [21].

Connection Setup: Once the master has found devices within its range, it can establish a connection to a device (i.e., paging). For a device to be paged, the master uses the slave device address to calculate the page-hopping sequence. To synchronize the phase in the sequence, the master uses the estimate of the slave's clock from the inquiry response, if available. To synchronize the phase, the master node performs a similar procedure as the inquiry procedure but uses the *page-hopping sequence*. The potential slaves periodically (i.e., $T_{\text{page\_scan}}$) enter the page scan state and stay there for the page scan window ($T_{w\_page\_scan}$) to search for paging packets. The slave PSRM (i.e., the scan frequency) can be either $R1$ ($T_{\text{page\_scan}} \leq 1.28$ s) or $R2$ ($T_{\text{page\_scan}} \leq 2.56$ s). The slave's setting (i.e., $R1$ or $R2$) is informed to the master via the PSRM field of the inquiry response packet. Since the paging interval should be greater than the page scan interval, the master sets the number of packet train repetition N_{page} based on the slave's configuration, i.e., $N_{\text{page}} \geq 128$ for $R1$ and $N_{\text{page}} \geq 256$ for $R2$. The master then starts sending the packet train N_{page} times. After receiving the paging packet, the slave immediately returns the response packet (without backoff). Finally, the master then responds with its FHS packet (master response), and the slave sends back an acknowledgement. As a result, a connection is established, and both master and slave begin to use the connection-hopping sequence defined in the master's FHS packet.

Power Class: A Bluetooth device is classified into three power classes: Class 1, Class 2, and Class 3 (100-, 10-, and 1-m radio range, respectively). Portable devices (PDA, smart phones, etc.) typically use Class-2 interfaces, and commercial

Bluetooth-based content distribution systems, such as BlueCasting [1] and BlueBlitz [2], use Class-1 interfaces. The output powers of Class 1, Class 2, and Class 3 must be in the range of [1 mW (0 dBm), 100 mW (20 dBm)], [0.25 mW (−6 dBm), 2.5 mW (4 dBm)], and [1 mW (0 dBm), N/A], respectively. The nominal output power is only defined in Class 2 as 1 mW (0 dBm). It is mandatory for Class-1 devices but optional for Class-2 devices to implement power control. Based on the received signal strength indication (RSSI), devices exchange messages (via the LMP) to adjust output power.

III. Measurement Study of Bluetooth-Based Content Distribution

In this section, we evaluate the Bluetooth-based content sharing system through measurement. We are mainly interested in measuring *peer discovery/connection setup latency* and the *data rate of a mobile user*. Peer discovery latency denotes the time to discover a random node. Connection latency is the time to make a connection to a discovered peer. The efficiency of the connection is measured through the download throughput. These performance metrics may depend on various factors, i.e., the class of Bluetooth devices (i.e., Class 1 or 2) and versions (i.e., Bluetooth 1.1, 1.2, or 2.0 EDR), the speed of the users, the distance between users, and WiFi interference. We measure the impacts of these variables using the metrics of interests. In this section, after describing the experimental setup, we first show the results of peer discovery/connection latency. We then present the downloading throughput of a mobile user, followed by the impact of WiFi interference. Finally, the downloading performance in a real environment is presented.

A. Measurement Setup

Measurement Hardware: Evaluating mobile users is challenging since it is nontrivial to control the speed of users. Thus, the key ingredient is the ability to control the speed of mobile users, so that we can repeat the experiment multiple times to accurately measure the performance. To this end, we use Amigobot, which is a programmable robot that can travel at a speed of up to 2.2 m/s. Amigobot can wirelessly be controlled in a remote machine using a 900-MHz radio modem pair called *AmigoWireFree*. A mobile user is emulated by an Amigobot carrying a laptop on its top. In the experiment, the speeds of the Amigobot are set to 0.5, 1.0, and 1.5 m/s (slow, moderate, and fast walking speeds, respectively), and our speed-measurement results confirm that it can achieve the specified settings of interest. Note that Amigobots are used to measure data rates under various scenarios (see Sections III-C and D); for peer discovery/connection setup (see Section III-B), we use static scenarios. The laptop used is Dell Latitude D610 with a Pentium M 770 processor (2.0 GHz) and 512-MB random access memory.

The following Bluetooth dongles are used for experiments. In the case of Class-2 devices, we use Belkin F8T003v (CSR chipset, Bluetooth v1.1), Bluetake BT009Si (Silicon Wave, Bluetooth v1.2), and Belkin F8T013V (Broadcom chipset, Bluetooth v2.0 EDR). Since most commercial Bluetooth-based content distribution systems such as BlueCasting [1] and BlueBlitz [2] use Class-1 interfaces, we also experiment with a Class-1 device, i.e., Belkin F8T012V (Broadcom chipset, Bluetooth v2.0 EDR), to see the potential benefits of its high transmission power for data transfer to Class-2 devices. Note that a long-range transfer is only feasible between Class-1 devices, because Bluetooth requires a bidirectional link for handshaking.

Measurement Environment: Unless otherwise mentioned, the experiments were carried out in the second level of an underground parking lot to exclude external factors, such as WiFi interference and obstacles (i.e., human). There was no physical interference of human/vehicles, because the experiments were performed late at night.

Measurement Software: To measure the metrics of interests, we develop a measurement tool using BlueZ v3.7 [22]. BlueZ is one of the most popular Bluetooth host protocol stack implementations and is included in the official Linux kernel. BlueZ implements core protocols (e.g., L2CAP and RFCOMM) and provides the BSD socket interfaces. The software is used for peer discovery, connection, and data transfer. On one side, the measurement software operates as the master node, and on the other side, it operates as the slave node. Peer discovery is carried out using the *hci-inquiry* function by the master node. The duration of inquiry (*inquiry length*) is one of the parameters of *hci-inquiry*, and its unit is 1.28 s.[1] The actual data transfer is realized through L2CAP sockets. L2CAP is a connection-oriented protocol that sends an individual datagram of fixed maximum length. The default transport policy is to retransmit until success or total connection failure, thus providing a *reliable* point-to-point connection.

The overall procedure can be summarized as follows: The master node first calls *hci-inquiry* to discover peers. The number of inquiry attempts is logged to measure the discovery latency. Upon finding a node, it tries to make an L2CAP socket connection to the peer. The latency of the L2CAP *connect* function is logged to measure the connection latency. As previously described, by setting the connection timeout as 3 s each, the connection attempt takes less than 3 s. The number of connection attempts is also logged. Dummy data with a chunk (packet) of 2300 B are continuously transmitted until the connection is lost. Hereafter, we refer to a chunk of 2300 B as a *packet*. For every packet transfer, the master logs the transmission power level (*hci-read-transmit-power-level*). The slave node sets on the inquiry and page scan and listens to a specific port to accept an L2CAP connection. Once the connection is created, the slave starts receiving packets and logs link quality (*hci-read-link-quality*) and RSSI (*hci-read-rssi*) information. For every 500-ms period, it logs the total amount of received data and the data throughput. In addition, the slave node logs HCI event messages using *hcidump*, which is a packet-snooping program for Bluetooth. As we will see later, *hcidump* at the receiver side allows us to infer the current packet type for data transfer. This information is managed by the LMP in Bluetooth and is not revealed to the upper layer.

[1]The inquiry window size is inquiry length $\times$ 1.28 s.

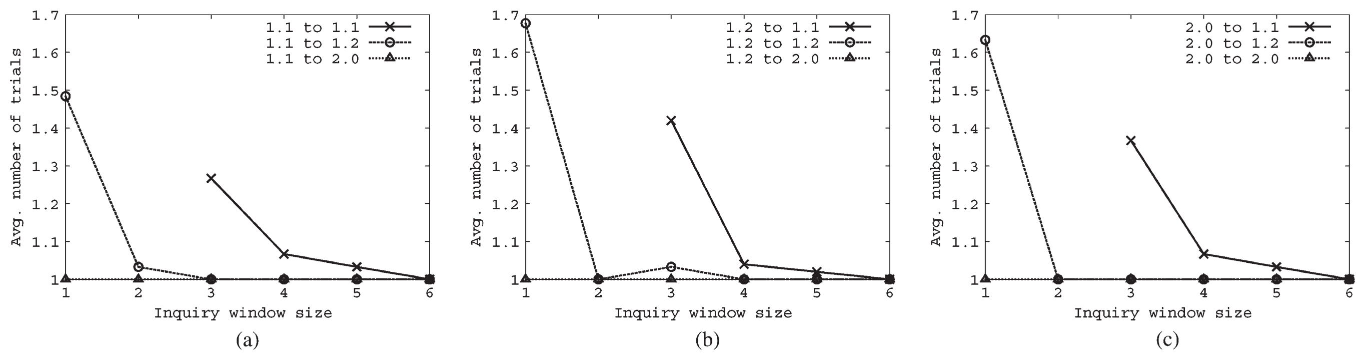


Fig. 3. Peer discovery as a function of inquiry length with various Bluetooth versions. (a) Bluetooth v1.1. (b) Bluetooth v1.2. (c) Bluetooth v2.0.

B. Peer Discovery/Connection Setup Latency

Peer discovery and connection latencies are the key components of Bluetooth-based content distribution. Given limited contact duration, the aforementioned overheads determine the useful time for the actual data transfer. In this section, we investigate various parameters impacting the performance of these metrics, i.e., inquiry length, distance between two peers, and Bluetooth versions. First, we show the impact of inquiry length in the range of [1, 6] with different Bluetooth versions. Recall that the unit of the inquiry window is 1.28 s, e.g., inquiry length 1 is equal to 1.28 s. If the inquiry fails, then the master node tries again. We measured the average number of such trials to discover a peer. Upon discovery, we measured the connection setup latency. To exactly measure the impact of distance, we use *static* scenarios where the distance between two peers was set from 0 to 20 m with a gap of 5 m. Bluetooth v1.1, v1.2, and v2.0 EDR were used. For each configuration, we ran experiments 50 times to get the average value.

Fig. 3 shows the average number of trials for various Bluetooth versions. Bluetooth v2.0 devices can successfully be discovered with inquiry length 1, and v1.2 devices take less than 2. This is mainly due to the use of interlaced scan operations introduced as of Bluetooth v1.2. In the interlaced inquiry scan, scanning one inquiry packet train is immediately followed by scanning the other train. The probability of missing an inquiry packet train is thus negligible [21], but it is still possible for the following reason: The specification recommends random backoff before a node returns an inquiry response, mainly to reduce the chances of collision when multiple devices have opened scan windows and an inquiring device begins transmitting inquiry packets. If the scanning interval is larger than 1.28 s, the range of [0,1023] is used for backoff. Otherwise, the range of [0,127] is used. Since the average backoff interval is one half of the maximum interval, the probability of failure with inquiry length 1 is given as $P[failure|T_{\text{w\_inq}} = 1, T_{\max\_bf} = 1024] = 512 \times 0.625\ \text{ms}/1.28\ \text{s} = 0.249$ and $P[failure|T_{\text{w\_inq}} = 1, T_{\max\_bf} = 128] = 64 \times 0.625\ \text{ms}/1.28\ \text{s} = 0.03$. From the figures, we see that the implementation may vary by vendors: Bluetooth v1.2 may use $T_{\max\_bf} = 1024$ as the maximum backoff window size, and Bluetooth v2.0 does not implement any random backoff.

Unlike other Bluetooth versions, Bluetooth v1.1 requires that the inquiry length be at least 3. In our experiments, we tried smaller inquiry lengths (i.e., 1 and 2), but discovery was not successful in most of the trials. Theoretically speaking, since inquiry lengths 1 and 2 show about 0.36 and 0.48 of discovery probabilities according to [21], the discovery process could be modeled using geometric distribution, assuming that each trial is independent. On average, by repeating a trial three times with inquiry length 1, we should be able to discover a peer. However, according to our finding, this is not true with a small inquiry length. In practice, we found that, for Bluetooth v1.1, we need at least inquiry length 3. Bluetooth v1.1 scans only a single train, and missing a train incurs 2.56 s (i.e., inquiry length 2) of penalty. We suspect that our tested Bluetooth device may always start with the same train, thus requiring at least inquiry length 3. The question is then whether we should repeat an inquiry procedure with a small inquiry length, e.g., 3, or try with a bigger inquiry length. For the sake of analysis, we assume that each trial is independent of one another if the inquiry length is greater than or equal to 3. Then, from Fig. 3, we can calculate the average latency for each Bluetooth version, i.e., the average number of trials is multiplied by the inquiry length. Let us find the optimal inquiry length to discover each Bluetooth version, i.e., by comparing the average latency of the plots with "1.1/1.2/2.0 to v1.1" in Fig. 3. For instance, for "v1.1 to v1.1," the latency with inquiry lengths 3 and 4 is 4.83 and 5.47 s, respectively.[2] Thus, we conclude that the optimal inquiry length to discover Bluetooth v1.1 and v1.2 or higher is given as 3 and 1, respectively.

We then measure the impact of distance on discovery latency. Fig. 4 shows the average number of trials as a function of distance. To discover Bluetooth v1.1 and v2.0 devices, we use inquiry lengths of 3 and 1, respectively. The distance is not a critical factor for device discovery. It confirms the fact that the inquiry procedure is robust, because an inquiry packet is repeatedly sent, and a small size inquiry packet has low packet error rate (PER).

Finally, we present the results of connection latency. Again, connection latency is the time for a node to connect to a discovered peer. Fig. 5 shows the results as a function of distance and Bluetooth versions. Surprisingly, the latency between Bluetooth v2.0 devices is twice larger than that between Bluetooth

[2]The average number of trials for inquiry lengths 3 and 4 is 1.07 and 1.25, respectively. Since the unit inquiry length takes 1.28 s, the average latency with inquiry lengths 3 and 4 is $1.07 \times 1.28 \times 4 = 4.83$ s and $1.26 \times 1.28 \times 3 = 5.47$ s, respectively.

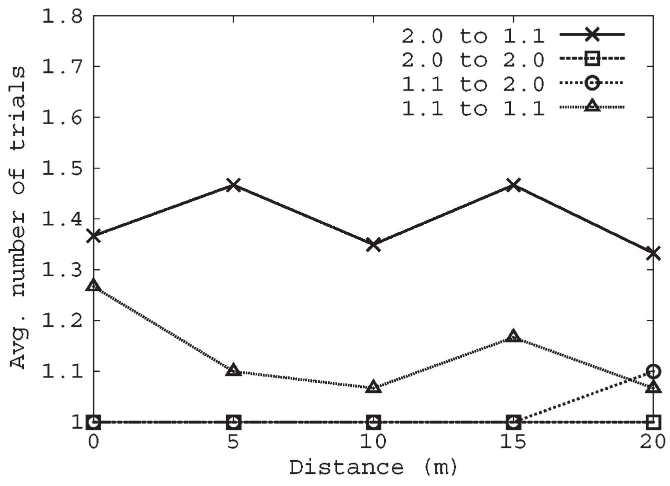


Fig. 4. Average number of trials as a function of distance.

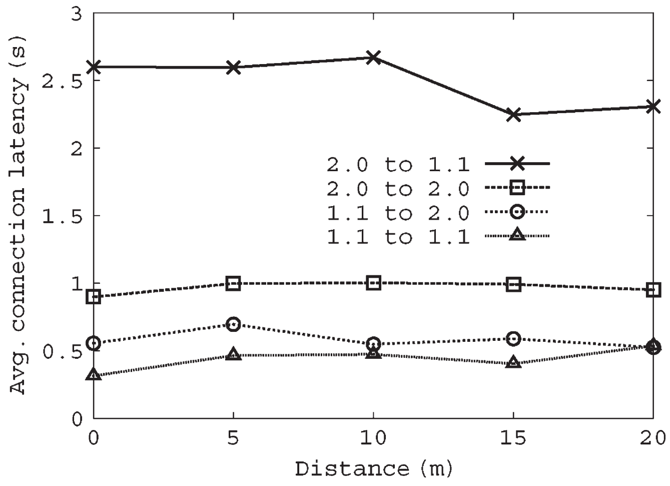


Fig. 5. Average connection latency as a function of distance.

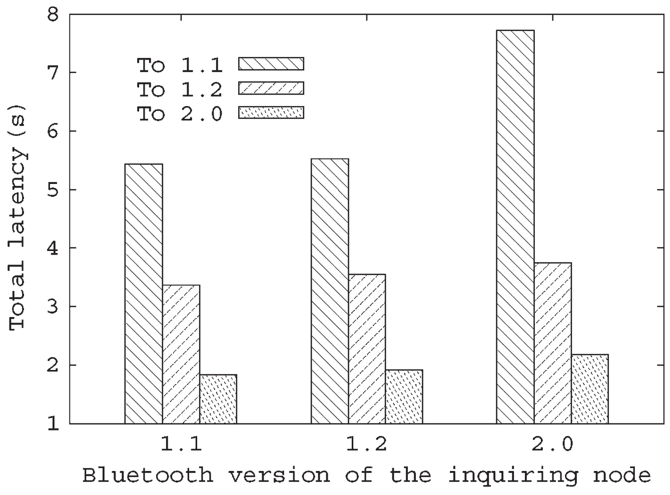


Fig. 6. Total latency. Discovery latency + connection setup latency.

v1.1. The latency for a Bluetooth v2.0 device to connect to a Bluetooth v1.1 device takes five times longer than that for the other direction. Similar to inquiry, the connection latency is not sensitive to distance. Fig. 6 shows the total latency for discovery and connection. In our scenarios under consideration, it takes much longer to discover and make a connection to a Bluetooth v1.1 device.

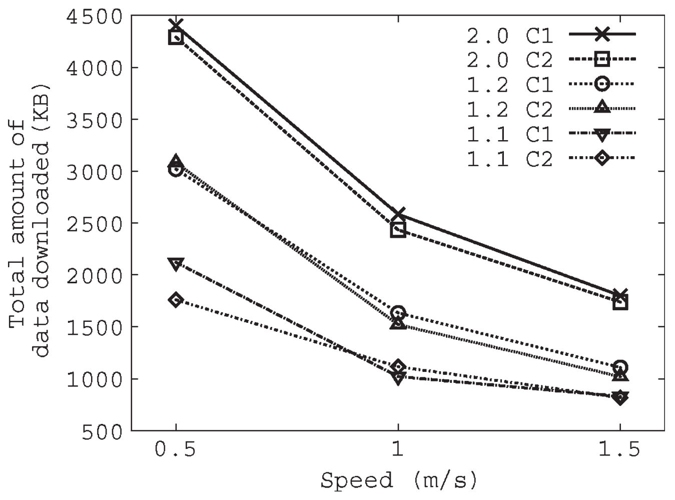


Fig. 7. Downloaded data size.

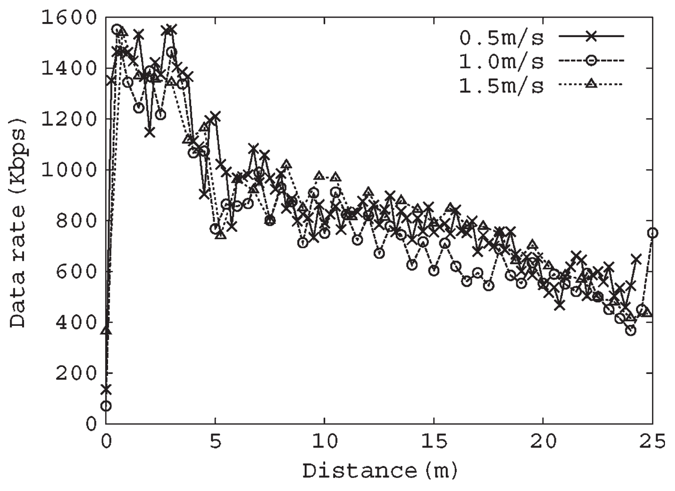


Fig. 8. Data rate of a 2.0 user (C2).

C. Download Throughput of Mobile Users

We measured the downloading throughput of a mobile user as follows: A static node as a master transferred data to a mobile user that is initially located at the same place. As described before, the mobile user was emulated using an Amigobot. We programmed the robot to travel up to 25 m at the speed of 0.5, 1.0, and 1.5 m/s to mimic slow, moderate, and fast walking speeds, respectively. The impact of various Bluetooth versions was explored by testing different Bluetooth versions at the mobile user side. We used a Class-1 Bluetooth v2.0 device to investigate the benefits of high transmission power at the sender side. We ran each configuration five times to calculate the average.

Fig. 7 shows the average of the total amount of data received while the Amigobot travels 25 m. It shows that a user moving at a moderate speed (i.e., 1 m/s) can download several megabytes. As the speed increases, the download data size decreases. The decrement is nonlinear since, for a given fixed distance, the traveling time is inversely proportional to speed (i.e., 0.5 m/s: 50 s, 1 m/s:25 s, 1.5 m/s:16.6 s); the contact duration is critical

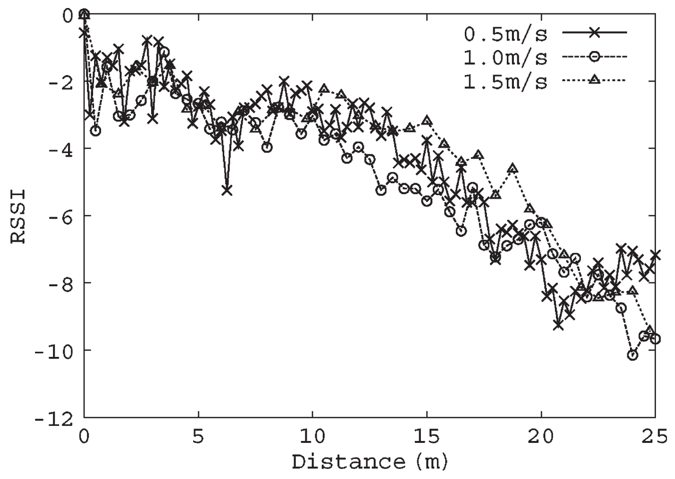


Fig. 9. RSSI of a 2.0 user (C2).

in the mobile environment. Fig. 8 shows the data rate as a function of distance with different speeds, respectively. The results show that the overall trend of the data rate is independent of the speed and is a function of distance. Due to the lack of space, we will not present the results of Bluetooth v2.0 Class-1 results, because the overall results are quite similar to Class-1 results, except that it shows higher RSSI on average. High transmission power (100 mW versus 1 mW) improves the performance by less than 5%, which validates the importance of a *symmetric* link in Bluetooth; Bluetooth protocol operations are bidirectional (e.g., peer discovery, data transfer, etc.). However, later, we will show that a Class-1 device may bring improvement in the presence of WiFi interference, at the cost of increased interference to WiFi.

Fig. 9 shows that the RSSI value gradually decreases with distance. On the other hand, the data rate sharply drops after passing by around the 5-m mark. This sharp drop is counterintuitive, because the bit error rate (BER) rather gradually decreases with distance [23]. We find that this is mainly due to CQDDR, which is the autorate control protocol in Bluetooth. The specification states that the quality measurement of a link at the receiver side can be used to dynamically control the packet type transmitted from the remote device to improve throughput (which is closely related to FEC and ARQ). However, the incorrect choice of a packet type may result in a significant performance loss, as noted by Chen *et al.* [16]. Therefore, the CQDDR policy plays a key role in determining the throughput, particularly in mobile environments. The results of Bluetooth v1.1 and v1.2 in Fig. 7 show that different policies may bring considerable throughput difference.

Let us further investigate the behavior of CQDDR in our tested Bluetooth devices. CQDDR is implemented in the LMP. The receiver side LMP can ask the sender side LMP to transmit packets using a preferred packet type. Although there is *no* HCI function that can access the current packet type being used, the current packet type can be *inferred* at the receiver side as follows: The L2CAP layer segments an L2CAP packet by ACL_MTU size, which can be read from the Bluetooth device. For example, given 1017 B of ACL_MTU (the default value of the Broadcom chipset), 2300 B of an L2CAP packet (+4 bytes for L2CAP header) is segmented into two 1017-B packets and one 270-B packet. Each segmented packet is attached with a 4-B ACL header (i.e., 1021-B and 274-B packets). A series of L2CAP segment packets are then sent to the lower layer. Bluetooth baseband processes those packets: If the size of an incoming packet is larger than that of the current packet type, a node fragments the incoming packet. For instance, in the preceding example, if the current baseband packet type is 3-DH5, which can load up to 1021 B, the segmented packet with size 1021 B is directly sent using a 3-DH5 packet. If the current packet type is 2-DH5 (maximum of 367 B), the baseband layer segments the payload, i.e., by generating two 367-B packets and one 283-B packet. The L2CAP layer of the receiver will reassemble the payload. Thus, by reading HCI event messages at the L2CAP layer, we can determine the current packet type. In our experiment, we use *hcidump* to log events. Due to the lack of space, we only present our main findings. First, the tested Bluetooth v2.0 devices start with 3-DH5 and then change to 2-DH5 and 2-DH3 as the link quality degrades. Second, the tested Bluetooth v1.1 devices use DH5 by default, and as the distance increases (i.e., after passing, on average, 10 m), it changes the packet type to DM5. Third, the tested Bluetooth v1.2 devices continue to use DH5, regardless of the distance.

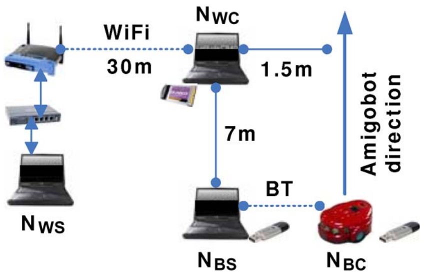


Fig. 10. WiFi interference test setup.

D. Impact of WiFi Interference

Both Bluetooth and IEEE 802.11 WiFi (i.e., 802.11 b/g) operate in the same frequency spectrum, i.e., the 2.4-GHz industrial, scientific and medical (ISM) band. Thus, it is expected that the performance of these devices is adversely affected by the presence of one another. In view of this, the Coexistence Task Group 2 (TG2) of IEEE 802.15 has included an AFH mechanism. AFH considers the channel condition and dynamically changes the hopping frequency, thus enabling coexistence with other devices in the 2.4-GHz ISM band. AFH consists of two steps: 1) channel classification and 2) adaptive control protocol. Channel classification keeps the list of "good" and "bad" channels based on the channel quality. This information is then exchanged through an adaptive control protocol. Given this, the AFH kernel chooses a set of hop frequencies to use by avoiding as many bad channels as possible. Note that, if the number of "good" channels is less than 15 (i.e., the minimum number of channels that the Federal Communications Commission requires Bluetooth to hop over), some bad channels would still be used.

We evaluate the throughput of mobile Bluetooth users in the presence of WiFi interference. The system configuration is

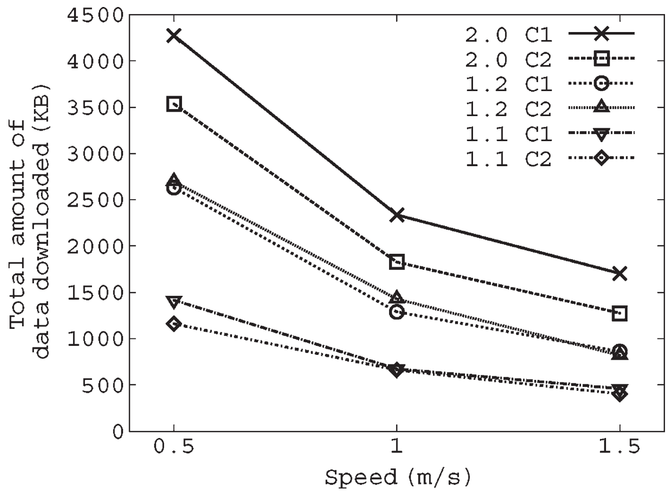


Fig. 11. Downloaded data size with interference.

shown in Fig. 10. Note that there was no other WiFi interference in our experiments. Recall that we performed the experiments on the second level of an underground parking lot to exclude external factors. We use a Linksys Wireless-G router, which is tuned to channel 8. Node N_{WC} with an Orinoco Gold 802.11b PCMCIA card is configured as a WiFi client located 30 m from the router. Both the wireless router and N_{WS} are directly attached to the hub. WiFi interference is induced using *iperf*, which is a bandwidth-measuring tool [24], by generating 4 Mb/s of User Datagram Protocol traffic from N_{WC} to N_{WS}. For each experiment, the bandwidth of N_{WS} is logged to show the impact of Bluetooth on WiFi. Node N_{BS} is located 7 m from N_{WC} and sends dummy data using a Bluetooth connection to the mobile node N_{BC}.

Fig. 11 shows the results of the total amount of downloaded data. Compared with that in Fig. 7, WiFi interference incurs throughput loss up to 30% in the scenarios under consideration. AFH is supposed to effectively circumvent this situation in both Bluetooth v1.2 and v2.0, but the loss is prevalent in both cases. To see this, we checked the AFH channel map via *hci-read-afh-map*. It returns 79 1-bit fields that represent the state of the hop sequence specified by the most recent AFH message exchange by the LMP. If channel n is used, the corresponding bit is 1; otherwise, it is 0. Our tested devices used all the available channels, even with WiFi interference. Interestingly, when we tested them in a static environment, it took about 30–60 s to detect interference. Although the choice of the channel estimation method is vendor specific, it is generally believed that some common methods such as PER or BER are used. However, these methods generally are on a channel-by-channel basis and thus require a longer response time. Moreover, the interference learning is on a *per-session* basis in our tested devices. A session must last at least that period of time to detect it, and the learned information is purged after a session is closed. Thus, AFH is not effective in mobile environments due to its long response time, as opposed to short contact duration.

From the figure, we also see that, unlike that in Fig. 7, a Class-1 device brings more than 20% of throughput gain for Bluetooth v2.0, but the performance of other versions is merely improved. When we plot the rate over distance for Bluetooth v2.0 Class 1 and Class 2 with WiFi, we observe that the impact of WiFi on a Class 2 device results in a drastic throughput decrement as it approaches to N_{WC}. On the other hand, Class 1 is fairly resilient to WiFi interference due to its high transmission power, which can adversely affect the performance of WiFi devices.

Note that the LMP also uses a power control scheme: If the received signal strength differs too much from the preferred value, then it may request an increase or a decrease in the TX power. Most products implement power control to save energy although it is not mandatory to implement the function. The current transmission power can be read by *hci-read-transmit-power-level*. Interestingly, we were not able to observe any power control working during the experiments. We examined tested devices to see whether power control is working or not: We located two laptops close by and initiated data transfer. After passing more than 30 s, the transmission power level gradually decreased. Note that the RSSI values were quite stable.

IV. Discussion

Our experimental results confirm that resource discovery is time/energy consuming, and a mobile user may experience a considerable throughput drop. In this section, we discuss strategies that can improve the performance of file swarming operations (i.e., resource discovery/downloading). In the extended version of this paper [25], we validate the proposed strategies via extensive experiments.

A. Cooperative Resource Discovery

The standard Bluetooth stack includes a service discovery protocol (SDP) that was proposed by the Bluetooth Special Interest Group (SIG). SDP provides a simple discovery mechanism based on successively requesting service classes or service attributes (i.e., file ID) from all devices in range. Since Bluetooth supports broadcast within a piconet, a node must form a piconet, i.e., the node as master should sequentially connect to each node as slave. It is, in fact, equivalent for a node to connect to each node and send a message one by one. As a result, the number of connection attempts per node will increase, and the chance of a useful contact will decrease, because each connection attempt will take at least several seconds, as shown in the previous section.

Peer discovery in Bluetooth can be used for broadcasting without connection setup. Recall that a master node broadcasts an inquiry packet and waits for inquiry response packets from the slave nodes. Sedov *et al.* showed that the 24-bit class of device (CoD) field of an inquiry response (FHS packet) can be used to expedite service discovery [26]. The CoD field has a variable format indicated using the format type field within the CoD. Bluetooth SIG assigned format type “00” to define the service category/device class (e.g., computer/laptop). In our protocol, we use format type “11,” which is reserved for future use. A node can recognize that other nodes are running our application by checking this format type.

In our scenario, content has a uniquely identifiable ID that is generated by applying a collision-resistant one-way hash function, e.g., SHA1. Since the length of the ID is longer than

22 bits (after excluding the 2-bit format type), we take the lower 22 bits of the ID. Each node sets the CoD field with the file that it is interested in downloading. Note that the CoD field can be set by an application using the *hci-write-class-of-device* command. After an inquiry, a master node can find neighbor nodes that are currently downloading files of interest. It then connects to one of those nodes to exchange pieces. Before the file exchange, these nodes can exchange a list of peers that they have collected so far to expedite file ID dissemination. Each entry includes a timestamp to denote when a node was discovered. We can set a time threshold to flush obsolete peers in the list, thus saving bandwidth.

In February 2007, the Bluetooth v2.1 standard draft was released. The major enhancement is the extended inquiry response (EIR) mode. The potential slave node can immediately send an EIR packet (maximum of 240 B) after returning the inquiry response. Thus, the EIR packet can contain more information, e.g., a file bitmap (of pieces that a node has).

B. Energy-Efficient Peer Discovery

Power consumption in a mobile device is a critical issue, and it is preferable to design an energy-efficient protocol that can preserve performance. Meier *et al.* [27] showed the statistics of power consumption per unit time in each Bluetooth operation, i.e., $C_{\text{Idle}} = 20$ mA, $C_{\text{Inquiry}} = 38$ mA, $C_{\text{Scan}} = 49$ mA, and $C_{\text{Data}} = 35$ mA. Although the scan operation is more expensive than an inquiry operation, the inquiry has to be repeated for a longer period of time. The total amount of consumed energy by inquiry is much *higher* than that by inquiry scan.

Bluetooth-based content sharing requires nodes to periodically alternate their role as master and slave, so that they can discover each other (i.e., P2P mode). Let us see how much energy a node spends in an hour to perform the P2P mode. Assume that the time spent for inquiry is the same as that for inquiry scan (i.e., 30 min). Thus, it spends $38\ \text{mA} \times 30\ \text{m} \times (1\ \text{h}/60\ \text{m}) = 19$ mAh for the inquiry and $\lfloor 30\ \text{m} \times (60\ \text{s}/1\ \text{m}) \times (1/1.28\ \text{s})\rfloor \times 49\ \text{mA} \times 11.25\ \text{ms} \times 10^{-3}\ \text{s} + [30\ \text{m} - \lfloor 30\ \text{m} \times (60\ \text{s}/1\ \text{m})(1/1.28\ \text{s})\rfloor \times 11.25\ \text{ms} \times 10^{-3}\ \text{s}] \times 20\ \text{mA} = 36458.7\ \text{mA} \approx 10.13$ mAh for the inquiry scan. During the idle period of inquiry scan (99.2%), most devices can change their state to the "standby" state (which consumes 2 mA), instead of the "idle" state [28]. Therefore, the actual energy consumption of the inquiry scan is approximately 1.21 mAh; this is why most devices (e.g., cellphones) typically stay in the inquiry scan mode. The cost of inquiry is significant, considering the fact that recent cellphones, such as the LG chocolate and Motorola MOTOKRZR K1m, are equipped with batteries that have less than 900 mAh.

We propose a simple solution to reduce the frequency of inquiry by using the concept of sociological orbits [29]. Sociological orbits are probabilistic mobility models where nodes move between a set of hubs, e.g., subway stations or bus stops. In Bluetooth-based content distribution, such hubs become information exchange bazaars [30] such that people with shared interests can cooperatively share content. Since it is less likely that information exchange happens in transit to hubs, we propose the *adaptive inquiry mode*, i.e., a node continues to stay in the *inquiry scan* state unless some other nodes in the *inquiry* mode wake it up by creating an actual connection. To be precise, in the very beginning, nobody is in the inquiry mode. Upon passing by an access point (AP), the node will be activated. Any node in the inquiry mode can then wake up others. If a node fails to create a connection or to find any nodes over a certain threshold, it then goes into the inquiry scan state to save energy. Instead of abruptly switching back to the scan state, the inquiry interval can dynamically be adjusted using the contact frequency (or recent activity), as proposed by Drula *et al.* [31].

C. Adaptive ACL MTU Size Selection

Let us illustrate the journey of application data in Bluetooth. Application data sent to the L2CAP layer are loaded into an L2CAP packet. The packet contains an L2CAP packet header with length and CID.[3] The L2CAP packet is then loaded into an HCI ACL packet. If it is larger than ACL_MTU, it is fragmented into multiple HCI ACL packets of size maximum transmission unit (MTU). Note that the ACL_MTU value is a fixed value read from the chipset when the BlueZ kernel module is loaded. An HCI ACL packet is then sent to the LMP in the Bluetooth chipset. If the current ACL packet type cannot accommodate an incoming HCI ACL packet, it will further be fragmented into multiple packets having the size of the current ACL packet type. The overall procedure is shown in Fig. 12. The problem happens when ACL_MTU is not a multiple of the current ACL packet size. We cannot fully utilize the channel, e.g., in the figure, and the shaded part of an ACL packet is wasted. Thus, we have to set the ACL_MTU value based on the current packet type of the LMP. However, Bluetooth reveals no information of the current ACL packet type (i.e., any information of CQDDR in the LMP), as shown in Section III-C. To illustrate this, consider the following example. An application sends a series of 1013-B packets. The L2CAP packet size is 1017 B (1013 B + 4 B L2CAP header). Assuming that ACL_MTU is 1017 B (which is a default value of the Broadcom chipset), there is no L2CAP level fragmentation. Given that the LMP starts with the packet type of 3-DH5 (maximum of 1021 B), an L2CAP packet is fully loaded into a 3-DH5 packet (1017 B + 4 B ACL header). Now assume that the packet type has changed to 2-DH5 (maximum of 679 B). A 1017-B L2CAP packet cannot be fitted into a 2-DH5 packet. At the baseband layer, the packet is fragmented and loaded into two 2-DH5 packets, i.e., 679 B + 338 B. Since the second packet cannot fully be utilized, this results in 24% throughput loss, i.e., 341 B out of 1358 B $(= 2 \times 679\ \text{B})$.

Although there is no HCI function available to read the current ACL packet type, in Section III, we show that the "receiver" can know the sender's packet type by reading HCI event messages at the L2CAP layer just as *hcidump*. As distance increases, the packet type gradually changes to a smaller size due to CQDDR (e.g., 3-DH5 to 2-DH5). Thus, we propose that the receiver notifies the sender of the packet-type change so that it can adjust the L2CAP payload size. The feedback overhead is minimal, because the response packet is small. Note that the

[3]CID is used to identify a logical channel endpoint of a device.

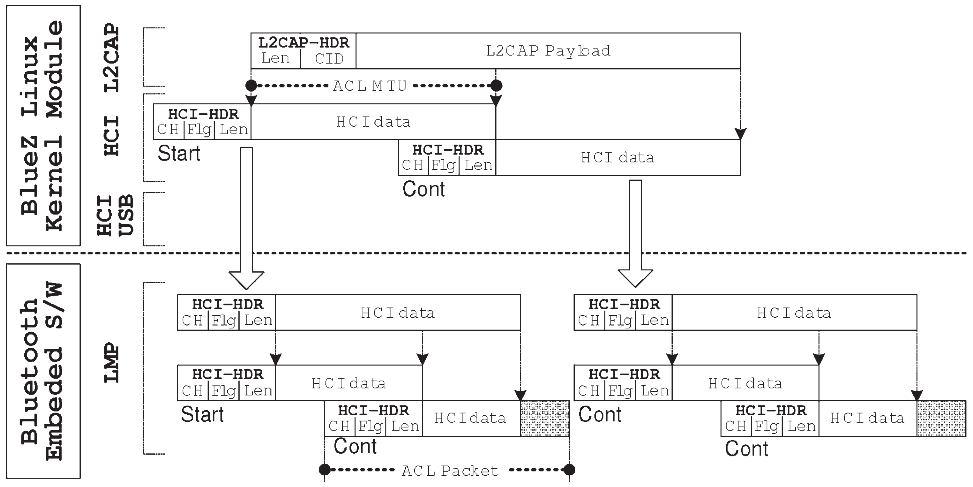


Fig. 12. Bluetooth stack overview. An L2CAP packet is fragmented into multiple HCI ACL packets of size ACL MTU. These packets are then transferred to the LMP, which is implemented in the Bluetooth chipset via HCI. The packet is sent without further fragmentation if the packet size is less than or equal to the size of the current HCI ACL packet; otherwise, it is fragmented into multiple packets. The shaded part of an ACL packet is wasted due to incorrect ACL MTU size.

proposed approach can be implemented in the BlueZ stack of Linux Kernel v2.6.15.1. During the L2CAP connection setup (*l2cap-do-connect*), mtu field of the L2CAP connection data structure (*l2cap-conn*) is initialized with ACL_MTU. This value is used to fragment the L2CAP packet in *l2cap-do-send()* (defined in $l2cap.c$). Thus, whenever receiving a feedback, a node updates mtu field (via a newly defined socket option).

V. Related Work

Bluetooth-based content distribution uses BT-AP to distribute content to the mobile users. It can be classified based on the cooperativeness of users. Noncooperative content distribution is based on a client–server model; BT-APs are the servers, and clients download files only from the APs. BlueCasting [1] and BlueBlitz Magic Beamer [2] are widely accepted proximity marketing systems such that they can identify Bluetooth users and deliver tailored messages. LeBrun and Chuah [10] proposed BlueSpot, which is a public-transit-based content distribution system that is accessible such that riders can opportunistically access data during their travel time via Bluetooth. In contrast, cooperative content distribution is based on a P2P model such that users can share the downloaded files from APs. Farkas and Bokos [11] proposed a push-based P2P content distribution model: A file was divided into blocks, and blocks were multicast through Bluetooth piconets. Given that nodes were synchronized and static, a ring overlay was formed to schedule parallel piconets to efficiently distribute blocks. Unfortunately, the scheme did not consider network dynamics such as mobility, churning, lack of synchrony, and intermittent connectivity. Jung *et al.* [13] implemented cooperative file swarming by optimizing peer discovery in Bluetooth to better utilize the short contact duration among mobile users. In the Haggle Project [33], Leguay *et al.* [12] proposed various content-distribution strategies to distribute a *small-size* file to a group of users in a large-scale urban network. Users can cooperatively relay a file to other interested users, and a set of mobile bridges (regardless of interests) can be used to further expedite delivery. However, none of the aforementioned works attempt to consider the actual characteristics of P2P Bluetooth communications in realistic environments (with mobility, heterogeneous Bluetooth versions, WiFi interference, etc.), even though that has a significant impact on the protocol design and evaluation. In this paper, we first evaluate Bluetooth communications in such an environment. The evaluation results are then used to examine various aspects of a system, e.g., resource discovery, limited bandwidth, resource availability, and heterogeneous Bluetooth versions/chipsets. Based on this, we propose techniques for efficient file swarming.

In spite of its popularity, the performance measurements of Bluetooth devices have not been thoroughly explored. Kasten and Langheinrich [7] evaluated their customized Bluetooth-based sensor device as a part of the Smart-its project and reported the issues of the power consumption and the inquiry latency. Similarly, Siegemund and Rohs [9] measured the performance of the inquiry procedure and then proposed a cooperative peer discover protocol. The measurement study by Leopold [8] is close to our work. The inquiry procedure and the throughput were evaluated using Bluetooth v1.1 devices. The author found that the inquiry was not sensitive to distance, and the latency distribution was centered on the train length. The measured throughput widely varied over distance and was much lower than their simulation results, which the author was not able to explain. Our work differs in the following aspects: 1) We explore the interoperability issues of various Bluetooth versions (v1.1, v1.2, and v2.0 EDR); 2) we measure the throughput of a “mobile” user; 3) we find the reason behind the throughput variation over distance; and 4) we explore the impact of interferences (e.g., WiFi/obstacles). Note that our inquiry results are consistent with those in [8] over all Bluetooth versions. Beaufour *et al.* [6] showed that the connection latency follows a long-tail distribution, i.e., quite a few take longer than 3 s. In contrast, we find that the connection setup rarely takes more than 3 s.

WiFi and Bluetooth coexistence issues are well documented in the literature [32], [34]. Shoemake [32] performed coexistence testing and showed that interference results in considerable throughput loss. Punnoose *et al.* [34] identified that effective bandwidth more rapidly degrades than the packet loss rate, owing to deferred transmissions caused by carrier sensing in WiFi. In contrast, the Bluetooth performance starts to rapidly degrade when the interfering WiFi signal is comparable to the desired signal level since Bluetooth does not use "carrier sensing." Shuaib *et al.* [35] evaluated the coexistence of 802.11g with Bluetooth. They showed that the shorter the distance between the Bluetooth and WiFi nodes, the lesser the impact on the throughput, mainly due to the power control in Bluetooth. Mander *et al.* [36] evaluated the AFH. They showed that the AFH improves the throughput by more than 30%. Unlike [8], they claimed that the average throughput does not significantly vary with distance. In this paper, we evaluate the impact of WiFi interference on "mobile" Bluetooth users. Contrary to the previous results [36], we find that, in the mobile environment, power control/AFH may not well potentially work due to the dynamics of RSSI and that the average throughput is dependent on the distance due to CQDDR.

VI. Conclusion

We have studied the practical implementation of P2P content distribution to Bluetooth users. We have measured the performance of Bluetooth operations, such as peer discovery, connection setup, and data throughput in dynamic environments with mobility, interference, and different Bluetooth versions. We have shown that resource discovery is time/energy consuming, even with the latest Bluetooth version, that mobile Bluetooth users experience considerable throughput drop not only from the imperfect autorate selection implementation in Bluetooth but from external interferences as well (obstacles and WiFi), and that the advanced features (power control and AFH) did not work well. We then discussed strategies that can improve the performance of resource discovery and downloading phases.

References

[1] BlueCasting. [Online]. Available: http://www.bluecasting.com
[2] BlueBlitz. [Online]. Available: http://www.blueblitz.com
[3] B. Cohen, "Incentives build robustness in BitTorrent," in *Proc. P2PECON*, Berkeley, CA, Jun. 2003.
[4] M. Conti, E. Gregori, and G. Turi, "A cross-layer optimization of Gnutella for mobile ad hoc networks," in *Proc. MobiHoc*, Urbana–Champaign, IL, Sep. 2005, pp. 343–354.
[5] U. Lee, J.-S. Park, J. Yeh, G. Pau, and M. Gerla, "CodeTorrent: Content distribution using network coding in VANETs," in *Proc. MobiShare*, Los Angeles, CA, Sep. 2006, pp. 1–5.
[6] A. Beaufour, M. Leopold, and P. Bonne, "The Bluetooth radio system," in *Proc. WSNA*, Atlanta, GA, Sep. 2002.
[7] O. Kasten and M. Langheinrich, "First experiences with Bluetooth in the smart-its distributed sensor network," in *Proc. PACT*, Barcelona, Spain, Oct. 2001.
[8] M. Leopold, "Evaluation of Bluetooth communication: Simulation and experiments," Dept. Comput. Sci., Univ. Copenhagen, Copenhagen, Denmark, Spring 2002.
[9] F. Siegemund and M. Rohs, "Rendezvous layer protocols for Bluetooth-enabled smart devices," *Pers. Ubiquitous Comput. J.*, vol. 7, no. 2, pp. 91–101, Jul. 2003.
[10] J. LeBrun and C.-N. Chuah, "Feasibility study of Bluetooth-based content distribution stations on public transit systems," in *Proc. ACM MobiShare*, Los Angeles, CA, Sep. 2006.
[11] P. S. Lóránt Farkas and B. Bakos, "A practical approach to multicasting in Bluetooth piconets," in *Proc. WCNC*, Las Vegas, NV, Apr. 2006.
[12] J. Leguay, A. Lindgren, J. Scott, T. Friedman, and J. Crowcroft, "Opportunistic content distribution in an urban setting," in *Proc. CHANTS*, Pisa, Italy, Sep. 2006, pp. 205–212.
[13] S. Jung, U. Lee, A. Chang, D. Cho, and M. Gerla, "BlueTorrent: Cooperative content sharing for Bluetooth users," in *Proc. PerCom*, White Plains, NY, Mar. 2007.
[14] L. McNamara, C. Mascolo, and L. Capra, "Media sharing based on colocation prediction in urban transport," in *Proc. MobiCom*, San Francisco, CA, Sep. 2008, pp. 58–69.
[15] AmigoBot. [Online]. Available: http://www.activrobots.com
[16] L.-J. Chen, R. Kapoor, M. Y. Sanadidi, and M. Gerla, "Enhancing Bluetooth TCP throughput via link layer packet adaptation," in *Proc. ICC*, Anchorage, AK, May 2002, pp. 4012–4016.
[17] K. Fall, "A delay tolerant networking architecture for challenged Internets," in *Proc. SIGCOMM*, Karlsruhe, Germany, Aug. 2003.
[18] U. Lee, S. Y. Oh, K.-W. Lee, and M. Gerla, "RelayCast: Scalable multicast routing in delay tolerant networks," in *Proc. ICNP*, Orlando, FL, Oct. 2008, pp. 218–227.
[19] J. C. Haartsen, "The Bluetooth radio system," *IEEE Pers. Commun.*, vol. 7, no. 1, pp. 28–36, Feb. 2000.
[20] Bluetooth SIG, Bluetooth Spec. v2.0, 2004.
[21] B. S. Peterson, R. O. Baldwin, and J. P. Kharoufeh, "Bluetooth inquiry time characterization and selection," *IEEE Trans. Mobile Comput.*, vol. 5, no. 9, pp. 1173–1187, Sep. 2006.
[22] BlueZ Bluetooth Protocol Stack for Linux. [Online]. Available: http://www.bluez.org
[23] A. Madhavapeddy and A. Tse, "A study of Bluetooth propagation using accurate indoor location mapping," in *Proc. UbiComp*, Tokyo, Japan, Sep. 2005, pp. 105–122.
[24] Iperf. [Online]. Available: http://dast.nlanr.net/Projects/Iperf
[25] U. Lee, S. Jung, A. Chang, D.-K. Cho, and M. Gerla, "Bluetooth-based P2P content distribution to mobile users," UCLA CSD, Los Angeles, CA, Oct. 2009. Tech. Rep.
[26] I. Sedov, S. Preuss, C. Cap, M. Haase, and D. Timmermann, "Time and energy efficient service discovery in Bluetooth," in *Proc. VTC*, Jeju, Korea, Apr. 2003, pp. 418–422.
[27] L. Meier, P. Ferrari, and L. Thiele, "Energy-efficient Bluetooth networks," Comput. Eng. Netw. Lab. (TIK), Swiss Fed. Inst. Technol. (ETH) Zurich, Zurich, Switzerland, Jan. 2005.
[28] 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 *Proc. ICWMC*, Bucharest, Romania, Jul. 2006, p. 13.
[29] J. Ghosh, S. Yoon, H. Q. Ngo, and C. Qiao, "Sociological orbit for efficient routing in intermittently connected mobile ad hoc networks," Univ. Buffalo, Buffalo, NY, Tech. Rep. TR-2005-19, Apr. 2005.
[30] M. Motani, V. Srinvasan, and P. S. Nuggehalli, "PeopleNet: Engineering a wireless virtual social network," in *Proc. MobiCom*, Cologne, Germany, Sep. 2005, pp. 243–257.
[31] C. Drulã, C. Amza, F. Rousseau, and A. Duda, "Adaptive energy conserving algorithms for neighbor discovery in opportunistic Bluetooth networks," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 1, pp. 96–107, Jan. 2007.
[32] M. B. Shoemake, "Wi-Fi (IEEE 802.11b) and Bluetooth coexistence issues and solutions for the 2.4 GHz ISM Band," Texas Instruments, Dallas, TX, Feb. 2001.
[33] Haggle Project. [Online]. Available: http://www.haggleproject.org
[34] R. J. Punnoose, R. S. Tseng, and D. D. Stancil, "Experimental results for interference between Bluetooth and IEEE 802.11b DSSS systems," in *Proc. VTC*, Rhodes, Greece, May 2001, pp. 67–71.
[35] K. Shuaib, M. Boulmalf, F. Sallabi, and A. Lakas, "Performance analysis: Co-existence of IEEE 802.11g with Bluetooth," in *Proc. WOCN*, Dubai, UAE, Mar. 2005, pp. 40–44.
[36] S. Mander, D. Reading-Picopoulos, and C. Todd, "Evaluating the adaptive frequency hopping mechanism to enable Bluetooth–WLAN coexistence," in *Proc. LCS*, London, U.K., Sep. 2003.

Uichin Lee (M'09) received the B.S. degree in computer engineering from Chonbuk National University, Jeonju, Korea, in 2001, the M.S. degree in computer science from the Korea Advanced Institute of Science and Technology, Daejeon, Korea, in 2003, and the Ph.D. degree in computer science from the University of California, Los Angeles, in 2008.

He is currently a Member of Technical Staff with Bell Laboratories, Alcatel-Lucent, Holmdel, NJ. His research interests include distributed systems, mobile wireless networking systems, and performance modeling/evaluation.

Sewook Jung received the B.S. degree in physics and the M.S. degree in computer engineering from Seoul National University, Seoul, Korea, in 1996 and 1998, respectively, and the Ph.D. degree in computer science from the University of California, Los Angeles, in 2007.

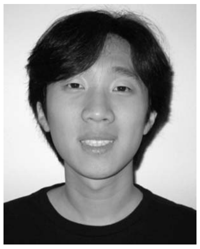
From 1998 to 2003, he was with Samsung Electronics. Since 2007, he has been a Staff Scientist with Broadcom Corporation, San Diego, CA. His research interests include wireless personal area networks (Bluetooth, ZigBee, ultra-wideband), Health-Net, ubiquitous computing, and network protocols.

Dae-Ki Cho received the B.S. and M.S. degrees in computer science from the University of California, Los Angeles (UCLA), in 2006 and 2008, respectively. He is currently working toward the Ph.D. degree in computer science with the Department of Computer Science, UCLA.

His research interests include wireless medical sensor networking, activity recognition, and cloud computing.

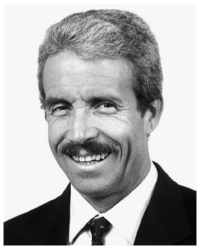
Alexander Chang received the B.S. degree in computer science and applied mathematics from the University of Washington, Seattle, in 2003 and the M.S. and Ph.D. degrees in computer science from the University of California, Los Angeles, in 2006 and 2008, respectively.

He is currently with Yahoo!, Inc., Burbank, CA. His research interests are wireless sensor personal networks and wireless network security.



Junho Choi received the B.S. degree in computer science and engineering and the M.S. degree in computer science from University of California, Los Angeles, in 2005 and 2007, respectively.

He is currently a Video Game Programmer with Heavy Iron Studios, Los Angeles.



Mario Gerla (F'02) received the Engineering degree from Politecnico di Milano, Milan, Italy, and the Ph.D. degree from the University of California, Los Angeles (UCLA).

He is currently a Professor of computer science with the Department of Computer Science, UCLA. At UCLA, he was part of the team that developed the early Advanced Research Projects Agency Network (ARPANET) protocols under the guidance of Prof. L. Kleinrock. From 1973 to 1976, he was with Network Analysis Corporation, Glen Cove, NY, where he helped transfer ARPANET technology to government and commercial networks. In 1976, he joined the UCLA Faculty, where he designed and implemented network protocols including ad hoc wireless clustering, multicast (on-demand multicast routing protocol and CodeCast), and Internet transport (TCP Westwood). He has lead the $12 M, 6-year Office of Naval Research MINUTEMAN project, designing the next-generation scalable airborne Internet for tactical and homeland defense scenarios. He is now leading two advanced wireless network projects under Army and IBM funding. His team is developing a vehicular testbed for safe navigation, urban sensing, and intelligent transport. A parallel research activity explores personal communications for cooperative networked medical monitoring (see www.cs.ucla.edu/NRL for recent publications).