An Empirical Study on the Performance of Bluetooth Scatternets

J.M. Cano-García, E. Casilari, E. Gonzalez-Parada
Departamento de Tecnología Electrónica at University of Málaga
Malaga, Spain
cano@dte.uma.es, ecasilari@uma.es, eva@dte.uma.es

***Abstract*—Bluetooth scatternet technology was conceived to provide support for ad hoc multihop networking applications. Many theoretical works in the literature have been dedicated to optimize the formation and operation of Bluetooth scatternets. However, the performance of a real scatternet strongly relies on the implementation of the Bluetooth stack in the interfaces of the network nodes. By analyzing the behavior of a simple scatternet of three nodes, this paper shows evidence of the scalability and delay problems that can appear when scatternets are deployed on existing hardware interfaces.**

Keywords-Bluetooth, scattenernet, packet delay, sniff mode

I. INTRODUCTION

Bluetooth (BT) is a leading standard for the deployment of communication services for PANs (Personal Area Networks), BANs (Body Area Networks) and WSNs (Wireless Sensor Networks). The flexibility of BT for the exchange of information has promoted the appearance of multiple BT-enabled applications and devices, ranging from handheld products (laptops, smartphones, tablets, etc.) to peripherals wireless sensors and gaming consoles. The basic networking solution of BT is the so-called Piconet, in which a single device assumes the central role of 'master' to coordinate the activity of the other nodes (the 'slaves'). A slave can exclusively communicate with its master and only after receiving a specific (poll) packet from the master. According to the BT specification, up to seven slaves and one master can simultaneously participate in a piconet. To extend the connectivity beyond these eight devices, BT defines the concept of scatternet. Scatternets are formed by the interconnection of two or more piconets. In this sense, a scatternet is created when a unit (either the master or one slave) of one piconet is admitted as a slave in other piconet. This node, which is a member of two piconets, will perform as a 'bridge' between the two networks. By utilizing this approach, several piconets can be joined together into a larger scatternet, expanding the coverage area of the network beyond the limited transmission range of a single piconet master.

BT specifications [1][2][3] do not impose the mode in which scatternets must be organized, structured or managed. Accordingly, problems such as routing, network formation, packet or scheduling in BT scatternets are still open issues which have been widely discussed by the literature on Bluetooth. However, most of these studies are theoretical or just validate their proposals by means of simulations. So, they often assume ideal characteristics in the capacities of the nodes, or they do not take into consideration the actual hardware implementation of the Bluetooth Controller in existing commercial BT interfaces.

By evaluating a simple real scatternet of three nodes and considering the two most basic topologies, this paper shows that actual BT devices present severe limitations to form scalable scatternets.

The paper is organized as follows: section 2 briefly comments some related literature on the topic. Section 3 describes the utilized experimental testbed while Section 4 shows and discusses the performed measurements. Finally, Section 5 recapitulates the main conclusions of the paper.

II. RELATED WORK

There is a vast literature dealing with the different issues related to BT scatternets (scheduling, routing, maintenance, formation, etc.). Many algorithms for diverse aspects of the management of scatternets have been proposed. Extensive surveys on network formation and routing in scatternets are for example presented in [4] and [5]. However, in the majority of these studies the scatternet topologies and algorithms are often evaluated in abstract terms, taking only into account metrics such as the connectivity, the average shortest path [6] or the time required to form the scatternet. Moreover, in these papers experimental evidence from a network prototype is normally not provided and conclusions are merely based on simulations where the whole protocol stack of BT is not emulated (or modeled in a very simplistic way). Some of these studies are devoted to the problems derived from the scheduling of the piconet switching in the bridge nodes. The paper in [7] presents an adaptive role switching technique to dynamically optimize the structure of a scatternet so that the transmission delay can be minimized. Similarly, the work in [8] proposes a new scheduling policy to overcome the link wastage due to the role/piconet switching in the scatternet bridge nodes. In [9] the queuing theory is employed to analytically characterize the performance of two Bluetooth piconets linked through a bridge point. To reduce the switch overhead experienced by the bridge nodes in a scatternet, authors in [10] suggest aligning the timing of all the piconets along each route. Authors in [11] call attention to the fact that many time slots in a scatternet are wasted by the masters when they are compelled to poll slaves that are inactive or with low traffic. To cope with this drawback they propose a scheduling algorithm that assigns

978-1-4673-2480-9/13/$31.00 ©2013 IEEE

slots to slaves and masters depending on the status of the traffic queues in the nodes, so that bandwidth can be shared fairly. Nevertheless, in all these works the methods are also validated through simulations.

Only a few studies in the literature on Bluetooth employ real testbeds to validate their technical proposals or to assess the applicability of commercial Bluetooth interfaces for certain networking applications. The paper in [12], for example, evaluates the suitability of BT technology for mobile location based message broadcasting.

There are even fewer works that analyze the feasibility of implementing scatternets in actual BT interfaces. In [13] authors investigate the performance of a particular policy for forming scatternets (*Bluepleiades*) when it is deployed on two distinct hardware platforms. By using an experimental testbed, the study focuses on the velocity of the nodes to discover neighbors and form the network. Results reveal that the behavior of an actual scatternet is poorer than that predicted by simulations. Authors justify this underperformance in a set of non mandatory features of the BT standard which are not implemented by the existing hardware. The work in [14] evaluates the throughput of a scatternet employed for a context-aware application to distribute information in a museum. The experiments are simply aimed at evaluating how the distance between the slave and the master affects the BT throughput in a two-node piconet. In fact, the authors have to emulate the performance of a scatternet as scatternet formation was not enabled in the employed BT stack at that time. In [15] a tree-oriented protocol for scatternet formation is described. The performance of the protocol is appraised in an experimental network. Again, as at the moment of the study the off-the-shelf hardware did not support the bridge operation in the nodes, the scatternet was emulated by using the hold power saving mode in the nodes.

The interesting study in [16] thoroughly examines the throughput in a real BT scatternet of up to 5 hops. The analysis is carried out for different scatternet topologies, background traffic and diverse measurement tools. Basing on their empirical results, authors observe that the switching times in the bridge nodes of the scatternet can degrade the performance of the scatternet severely. Moreover, the paper concludes that simple temporary 1-to-1 connections between the piconets enable a more effective mode for interconnecting piconets than a scatternet itself. Similarly, the throughput of the three typical topology configurations of a 3 node BT scatternet is also investigated in [17]. Basing on the analysis of the behavior of a FTP transaction, authors state that a bridge acting simultaneously as a master and a slave of two piconets can degrade the overall throughput. This is explained by the fact that this type of bridge asymmetrically distributes its activity among the networks. The same paper evaluates the performance of a two-hop BT scatternet (in terms of jitter and packet loss) when it is used to support a videoconferencing application. Nevertheless, in these two articles the packet delay is not considered as a QoS (Quality-of-Service) metric while the benefits of using a saving power mode to harmonize the node activity in the scatternet are not contemplated.

III. MEASUREMENT TESTBED FOR THE ANALYSIS

The measurement testbed, which has been depicted in Fig. 1, consists of three Personal Computers (PC) connected through a 100 Mbps Ethernet LAN (Local Area Network) switch. The PCs also incorporate a BT dongle with CSR BlueCore 4 firmware [18], implementing the version 2.1 of Bluetooth. We chose these devices as CSR is the most popular vendor of BT technology

As the Operating System in the PCs was Linux, the employed BT stack was BlueZ [19]. This stack, which supports the main core BT protocols and layers, includes an interesting set of network configuration tools and testing utilities.

All the communications in our testbed are based on UDP/IP connections. For this purpose, the PAN (Personal Area Network) BT profile is employed. This profile specifies how two or more Bluetooth devices can set an ad-hoc network, and how to access remote networks through access points. The main advantage of the PAN profile is that it enables an IP-based service. The PAN profile utilizes BNEP (Bluetooth Network Encapsulation Protocol). This protocol, inspired by Ethernet, was specifically devised for the transport of IP frames over Bluetooth. Thus, the PAN nodes (PANUs and NAP) can be addressed in a transparent manner from any IP network. By means of a Bluetooth PAN daemon (*pand*) provided by BlueZ, one of the computers is configured as a BT Network Access Point (NAP). Conversely, the other two terminals, which are programmed to perform as PANU (PAN Users), establish two different BT links with the NAP.

When the PANUs initiate the connection to the NAP, two separated piconets are created. Initially, the NAP assumes the role of the master in both piconets while the PANUs are configured as slaves by default. However these roles can be swapped with the *hcitool* utility in order to create the diverse scatternet topologies that will be evaluated. In any case, Linux allocates a different network interface (*bnep0* and *bnep1*) in the NAP for each BT link (with independence of the employed scatternet topology).

As it can be observed in Fig. 1, the two devices acting as PANUs have two different IP addresses, which are associated to the Ethernet and Bluetooth network interfaces, respectively. On the other hand, the NAP has three network interfaces: one for the Ethernet card (*eth0*) and two for the Bluetooth connections to the two PANUs (*bnep0* and *bnep1*). With the Linux *ifconfig* tool, a specific IP address is again assigned to each interface. Consequently, the NAP is a member of three networks (two Bluetooth piconets and an Ethernet LAN). So the NAP will behave as an IP router in the scatternet by communicating both BT PANUs.

The goal of the testbed is to estimate the delay of the UDP/IP packets between the PANU and the NAP in both uplink and downlink directions for different configurations of the scatternet topology.

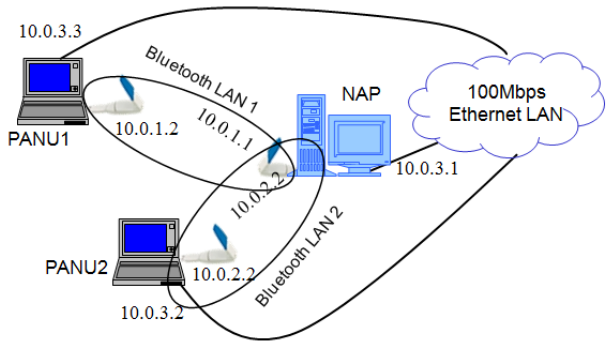

Figure 1. Evaluation Testbed.

As it is illustrated in Fig. 2, the routing process in one of the PANUs (PANU 1) is configured so that any packet coming from the NAP through the BT interface is forwarded back to the NAP via the Ethernet interface. Similarly, all the incoming Ethernet traffic is re-sent by the PANU to the NAP through the uplink BT connection. This re-configuration of the routing mechanism in the NAP is achieved with the *iptables* component included in the Linux Netfilter framework. *Iptables* offers a generic table structure that allows the creation of routing rules as well as it facilitates Network Address Translation (NAT). In our testbed, two *prerouting / postrouting* rules are added so that when a packet with an IP destination address 10.0.1.2 and a source address 10.0.1.1 is received by the PANU (via Bluetooth), the addresses are 'NATted' to 10.0.3.1 and 10.0.3.3, respectively (the IP addresses of the NAP and the PANU associated to the Ethernet interface). Consequently, the packet is forwarded to the Ethernet link by the IP layer of the PANU. By creating the complementary rules, the incoming Ethernet packets with a destination address 10.0.3.3 flowing from the NAP (10.0.3.1) are retransmitted via Bluetooth to the NAP with a source address 10.0.1.2 and destination address 10.0.1.1.

A specific software application for the measurements is developed and installed in the NAP. The application is in charge of injecting probe UDP packets trough one of the network interfaces of the NAP (Bluetooth or Ethernet). Every injected packet incorporates a sequence identification number and a timestamp describing the time in which it was generated. To estimate the uplink delay, the UDP packets are targeted to the IP address of the PANU associated to the Ethernet interface. Once they are delivered to the IP layer at the PANU, the datagrams are redirected to the IP address of the NAP that is linked to the Bluetooth interface. Upon receiving a packet, the application in the NAP computes the corresponding Round-Trip-Time (RTT) as the difference between the present time and the timestamp included in the payload.

The global delay due to the transmission in the 100 Mbps Ethernet LAN and the processing time of the protocol stack (i.e. the NAT operation) in the terminals is proved to be extremely low. We checked this point by substituting the BT link between the PANU and the NAP by an Ethernet crossover cable. Under this scheme, after sending a set of 10000 probe packets through a closed loop (using NAT), the mean and the 99.99th percentile of the computed delay were estimated to be 0.11 ms and 0.186 ms, respectively. Consequently we assume that the RTT that is measured when the BT connection is active is basically determined by the uplink transmission in the BT link between the PANU and the NAP. Thus, we consider this RTT as a reasonable estimation of the delay in the uplink direction of the BT connections. In a similar way, the downlink packet delay is also measured in the NAP by sending probe packets to the PANU through the BT interface (downlink direction). Now, these packets are sent back via Ethernet to the NAP, which will calculate the RTT.

By using these retransmissions, the packet delay is measured in the NAP without requiring any further synchronization among the nodes, just using a single clock (in the NAP).

All probe packets are sent on a periodic basis (with a mean period of 100 ms). To avoid a spurious synchronization of the transmissions with the system dynamics of the scatternet, the period was not set to a constant value. Hence, after emitting a probe packet, the next packet is delayed for a period randomly chosen between 80 and 120 ms.

The measurements for each considered scenario are based on a series of 10000 UDP packets. As it is the case of many applications of wireless sensor networks, the size of the packet is small and constant. In particular, every packet transports 10 data bytes plus 28 bytes due to the overhead (20 bytes of the IPv4 header and 8 bytes of the UDP header). In all cases, the traffic is generated in a single direction (uplink or downlink) and just in one of the two links that form the scatternet (or the piconet). This corresponds to the simplest case where there is not any background traffic in the opposite direction or in the other link of the network.

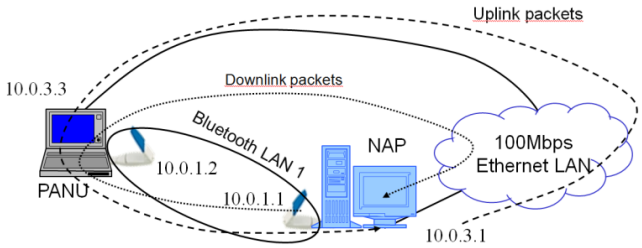

Figure 2. Measurement of packet delay for uplink and downlink directions.

The selected QoS (Quality of Service) parameters of the Link Manager Protocol are the default ones. Thus, according to the manufacturer's specifications, the poll time (T_{poll}) is set to a value of 40 BT slots (25 ms). This parameter (latency parameter of the CSR chipset) defines the maximum time between two consecutive transmissions of a BT device. Its importance lies in the fact that a slave will only transmit a packet after being polled by the master (i.e. after receiving a poll or a data packet from the master).

IV. RESULTS

Using the scenario and tools previously described, different tests have been conducted to investigate the performance of

several scatternet topologies under diverse conditions. This section presents some preliminary results.

Taking into account the roles that can be assumed in a network of three nodes, three simple scenarios (sketched in Fig. 3) are evaluated:

1. A piconet with a master and two slaves (no scatternet is formed). We use this scenario as a reference of the negative impact of the use of *scatternets* on the delay of BT connections.
2. A scatternet with an M-S/M-S topology, that is to say, with a bridging node (S/M) acting as a slave for a master (M) and as a master for a slave (S). For this topology we have to investigate the delay of the communications between the masters and slaves separately, as long as the structure of the network is not symmetrical.
3. A scatternet with an M-S-M topology in which the node acting as the bridge is the slave of two different masters. In this case, the symmetry of the topology allows assuming that the delay between the slave and one master will not depend on the master that is selected for the analysis.

The basic results (mean and median) for each configuration are summarized in Table 1. As expected, results show that the delay in the downlink direction (from the master to the slave) is always smaller than in the uplink direction. This is clearly motivated by the fact that the slave has to wait to be polled by the master before transmitting any packet. Conversely, according to the way in which actual BT chipsets are configured, the BT controller of the master can address the slave as soon as it receives a packet from the higher layers (without waiting the whole poll period).

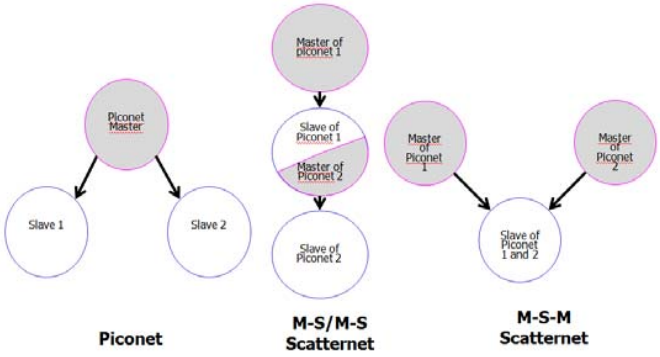

Figure 3. Evaluated Scatternet topologies.

As to the comparison between the performances of the piconet and the scatternet, the results indicate that in the case of the M-S/M-S topology, the 'hybrid' bridge S/M node exhibits a similar behavior to that of a piconet. In particular, the delay is slightly lower in both the uplink direction of the connection with its slave (S/M ← S) and in the downlink direction of the connection with its master (M →S/M). As the bridge node is obliged to poll its slave periodically, there exists a certain probability of losing the poll from its master. This probability is minimized in the router by remaining as a slave most of the time. This fact can explain a slightly higher delay in the uplink direction (M←S/M) of the connection between the bridge node (S/M) and its master. In addition, the hardware of the BT controller in the bridge requires some time to execute the role switching operation (moving from slave to master). This role swap could also account for the increase in the packet delay in the downlink direction from the bridge node to its slave (S/M→S).

On the other hand, the performance of the scatternet clearly degrades if the M-S-M topology is considered. In particular, the delay in the uplink direction between the bridging slave and its masters nearly doubles that of the piconet. The comparison of the histograms of the delay (see Figures 4 and 5) in the uplink direction reveals the existence of a multimodal distribution of the delay in the case of the M-S-M scatternet when compared with the scenario of the piconet. This can be justified by the fact that the slave is being alternatively polled by the masters without any specific synchronization between the two polling processes. This involves again a certain probability of not receiving the poll from a certain master because the bridge is listening to the other master. As the time as slave is equally shared between the masters, this probability of losing a poll packet is fairly higher than in the previous M-S/M-M topology. Whenever this poll loss occurs, a packet generated by the bridge node will have to wait at least another poll period (T_{poll}) before proceeding to the data transmission. The parameter T_{poll} (25 ms by default) can be set to a lower value (with a practical limit of 10 ms) but at the cost of increasing the battery consumption in the master. The modes in the histogram (see Fig. 5) clearly denote the impact of the polling mechanism on the delay experienced by the uplink packets in the M-S-M scatternet.

TABLE I. PACKET DELAY FOR UPLINK AND DOWNLINK DIRECTIONS

		Piconet (no scatternet)	Scatternet configuration		
			M-S/M-S		M-S-M
Direction	**Metric**		**M-S/M**	**S/M-S**	
Uplink *(M←S)*	***Mean***	15.6 ms	19.1 ms	16.6 ms	28.0 ms
	Median	15.4 ms	17.0 ms	16.2 ms	24.8 ms
Downlink *(M→S)*	***Mean***	3.5 ms	3.7 ms	6.0 ms	7.3 ms
	Median	3.2 ms	3.2 ms	5.4 ms	5.3 ms

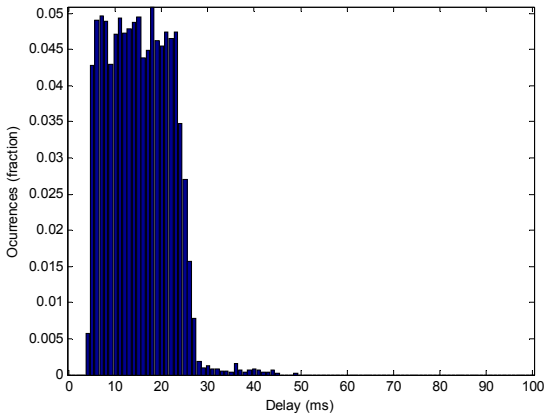

Figure 4. Normalized histogram of the delay for the uplink direction.

Piconet topology

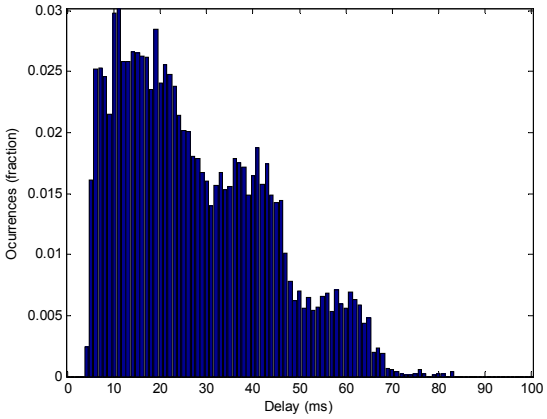

Figure 5. Normalized histogram of the delay for the uplink direction. M-S-M scatternet topology

A. *Utilization of the sniff mode*

Bluetooth enables different power saving modes (*hold*, *park* and *sniff*) to reduce the duty cycle of the BT nodes. When the sniff mode is employed, the master and the slave negotiate specific slots (*sniff* slots) where the packet exchange can begin. A Bluetooth slave in the Sniff mode listens to the piconet at regular intervals (T_{sniff}) for a short period during which it must be addressed by the master. The rest of the time the BT modules can remain in a low consumption state, which increases the battery lifetime.

From the point of view of the performance of a scatternet, the main benefit of using the sniff mode is that it obliges the master and the slave to synchronize their activity. This helps to reduce the typical losses of poll packets that take place in a scatternet because the bridge node cannot participate in two piconets simultaneously and, more specifically, when the bridge node acts as a slave for two different masters.

To evaluate the improvements of the use of the sniff mode in a scatternet, we repeated the measurements of the uplink delay with the M-S-M. In this case, the sniff mode is set to regulate the communications between the bridge node (slave) and the master to which the packets are targeted. The results of these new experiments, which were replicated for different values of T_{sniff} , are tabulated in Table 2. From these results we can observe that the sniff mode can have a beneficial effect on the achieved performance of the scatternet. The synchronization between the master and the slave imposed by the sniff mode helps to significantly reduce the delay in the scatternet if T_{sniff} is set to a small or a moderate value. The histogram in Fig. 6 shows that the multimodal distribution of the delay has disappeared, which denotes that the poll packets from the master are now conveniently received by the slave (bridge) node. Conversely, the dynamics of the waiting times set by the sniff mechanism only become the main source of the delay for high values of T_{sniff} (e.g. 100 BT slots, which correspond with 62.5 ms).

TABLE II. PACKET DELAY FOR UPLINK AND DOWNLINK DIRECTIONS USING THE SNIFF MODE. SCATTERNET CONFIGURATION M-S-M.

Direction	Metric	No sniff mode	Configured Sniff period (T_{sniff})			
			10 slots	20 slots	30 slots	100 slots
Uplink (M←S)	*Mean*	28.0 ms	9.18 ms	12.0 ms	14.8 ms	37.7 ms
	Median	24.8 ms	9.0 ms	11.9 ms	14.6 ms	37.3 ms
Downlink (M→S)	*Mean*	7.3 ms	7.1 ms	10.1 ms	13.5 ms	36.1 ms
	Median	5.3 ms	7.0 ms	10.0 ms	13.4 ms	35.7 ms

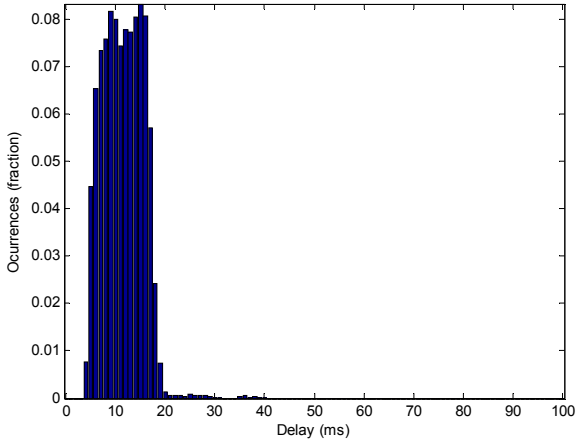

Figure 6. Normalized histogram of the delay for the uplink direction. M-S-M scatternet topology. Sniff mode is employed with a T_{sniff} of 20 slots.

CONCLUSIONS

According to the Bluetooth specifications, scatternets are purely conceptual as the standards do not provide any guideline to determine how a scatternet topology should be arranged. Thus, since the apparition of Bluetooth much research efforts have been devoted to a wide set of topics regarding the configuration, routing and formation of BT scatternets. In this sense, complex algorithms and policies have been proposed to optimize the performance of the scatternets. However, the results of most of these works are based on simulations. In many cases, the proposals assume certain characteristics in the nodes that are not normally implemented in the actual BT hardware. To the best of our knowledge, very few works in the literature are founded on the study of a real multihop ad hoc network testbed. In fact, still today, some commercial Bluetooth interfaces do not support scatternets properly. Moreover, the formation of scatternets is not even contemplated in the initial release of Bluetooth 4.0 Low Energy.

In this work we have shown the preliminary results of an empirical characterization of scatternets deployed on actual BT interfaces. The analysis has been focused on the delay experienced by UDP/IP datagrams in a single hop of a basic scatternet. Experimental results indicate that the scheduler in real BT node acting as a bridge in a scatternet simply follows a basic Round Robin (RR) policy. Thus, an important fraction of the transmission time is wasted because of the frequent piconet switching that are required to fulfill the typical polling process between BT masters and slaves. With this inefficient

scheduling policy, packet delay can be doubled (at least) per every hop when compared with the delay in a one-hop piconet. Consequently, we can conclude that present BT hardware is not prepared to build networking solutions based on multihop scatternets. Results also show that the use of power-saving modes (such as Sniff mode) can mitigate this problem as long as it synchronizes the activity of slaves and masters.

ACKNOWLEDGMENT

This work was partially supported with European, Spanish and regional public funds by Project No. TEC2009-13763-C02-01 and Andalusian Project P08-TIC4198.

REFERENCES

[1] Bluetooth SIG, "Bluetooth Core Specification Version 2.0+EDR". In *Specification of the Bluetooth System*. Bluetooth Special Interest Group, 10 November 2004.

[2] Bluetooth SIG, "Bluetooth Specification Version 4.0", Bluetooth Special Interest Group, December, 2009.

[3] Bluetooth SIG, "Bluetooth Specification Version 3.0 + HS," Bluetooth Special Interest Group, December 2009.

[4] K. Persson, D. Manivannan, and M. Singhal, "Bluetooth scatternet formation: criteria, models and classification," Ad-hoc Networks, No. 3, pp. 777-794, July 2005.

[5] R. M. Whitaker, L. Hodge, and I. Chlamtac, "Bluetooth Scatternet Formation: A Survey," Ad-hoc Networks, nº 3, pp. 403-450, July 2005.

[6] P.E. Engelstad, T.V. Do, and T.E. Jønvik, "Formation of Scatternets with Heterogeneous Bluetooth Devices," In Proc. of 3G-Wireless Conference 2003 (3GWireless'03), May 2003.

[7] C.Y. Chung, and H.R. Chang, "Adaptive Role Switching Protocol for Improving Scatternet Performance in Bluetooth Radio Networks," IEEE Transactions on Consumer Electronics, vol. 52, no. 4, pp. 1229-1238, November 2006.

[8] Y.I. Joo, T.J. Lee, D.S. Eom, Y. Le, and K. H. Tchah, "Power-Efficient and QoS-Aware Scheduling in Bluetooth Scatternet for Wireless PANs," IEEE Transactions on Consumer Electronics, vol. 49, no. 4, pp. 1067-1072, November 2003.

[9] J. Misic, and V.B. Misic, "Bridges of Bluetooth county: topologies, scheduling, and performance," IEEE Journal on Selected Areas in Communications, vol.21, no.2, pp. 240- 258, February 2003.

[10] Y. Liu, M. Lee, and T. Saadawi, "A Bluetooth scatternet-route structure for multihop ad hoc networks," IEEE Journal on Selected Areas on Communications, vol. 21, no. 2, pp. 229–239, February 2003.

[11] R.G. Reddy, and S. Bhatnagar, "An Efficient and Optimized Bluetooth Scheduling Algorithm for Scatternets," In Proc. of IEEE Advanced Networks and Tele-communication Systems (IEEE ANTS 2008), pp. 1-3, Dec. 2008.

[12] M. Aiello, R. de Jong, and J. de Nes, "Bluetooth broadcasting: How far can we go? An experimental study," In. Proc. of Joint Conferences on Pervasive Computing (JCPC 2009), vol., no., pp. 471-476, December 2009.

[13] C. Petrioli, C. Pierascenzi, and A. Vitaletti, "Bluetooth Scatternet Formation Performance: Simulations vs. Testbeds," In Proc. of Military Communications Conference (MILCOM 2006), pp.1-7, October 2006.

[14] J.C. Cano, D. Ferrández-Bell, and P. Manzoni, "Evaluating Bluetooth Performance as the Support for Context-Aware Applications," Telecommunication Systems, Vol. 28, Issue 3, pp. 333-347, March 2005.

[15] D.N. Kalofonos, and S. Asthana, "A Bluetooth scatternet formation and healing protocol for ad-hoc group collaboration," In Proc. of . 2nd International Conference on Broadband Networks (BroadNets 2005), vol. 2, pp.774-777, October 2005

[16] S. Jung; A. Chang, and M. Gerla, "Comparison of Bluetooth Interconnection Methods using BlueProbe," In Proc. of 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, pp. 1- 6, April 2006.

[17] F. Cuomo and A. Pugini, "A linux based bluetooth scaternet formation kit: from design to performance results", in Proc. of IEEE ICPS Workshop on Multi-hop Ad Hoc Networks: from theory to reality (REALMAN'2005), pp.43-50, July 2005.

[18] CSR, "Cambridge Silicon Radio Plc. BlueCore Bluetooth chipset," URL: http://www.csr.com.

[19] BlueZ (official Linux Bluetooth protocol stack), URL: http://www.bluez.org)