A Simple Adaptive MAC Scheduling Scheme for Bluetooth Scatternet

Changlei Liu and Kwan L. Yeung
Department of Electrical and Electronic Engineering
The University of Hong Kong
Hong Kong, PRC
Tel: (852) 2857-8493 Fax: (852) 2559-8738 E-mail: {clliu, kyeung}@eee.hku.hk

Abstract — **A simple adaptive MAC scheduling algorithm, called Gateway Oriented Scatternet Scheduling (GOSS), is proposed for data exchange in a Bluetooth scatternet. Unlike the existing scheme MDRP (Maximum Distance Rendezvous Point) that has a global superframe schedule shared by all gateways, the schedule used by each gateway is individually determined. Equal partition of the superframe schedule at a gateway to each connected piconet can thus be guaranteed, which enables a more robust performance than MDRP. In addition, GOSS allows a variable sized superframe at each gateway. To maximize its performance, the frame size can be dynamically adjusted according to the scatternet topology and traffic load in every predefined adaptation interval. Simulation results show that even a static GOSS prevails over MDRP. If the adaptation technique is used, further performance enhancement can be found.**

I. INTRODUCTION

Served as the baseline of IEEE 802.15 PAN (Personal Area Network) standard, Bluetooth [1] is promising to become the first commercial ad hoc network. Although it is initially designed for cable replacement, the desire for mobility coupled with the demand for ad hoc connectivity gives rise to the concept of *scatternet*, which consists of a collection of *piconets*. A piconet has a star-like topology with one Bluetooth device functions as a master and up to seven other devices function as its *slaves*. Different piconets can be connected to form a *scatternet*, via some common node, known as *gateway*. A gateway can be a common slave in all connected piconets, or it can be a master in one piconet and a slave in the rest (denoted as a master/slave gateway). Fig. 1 shows an example of Bluetooth *scatternet*. It consists of 5 *piconets* interconnected by two (common slave-typed) *gateways*, nodes 0 and 1. The construction, manipulation, and optimization of scatternets are very interesting problems but beyond the scope of this paper.

Data exchange within a piconet is master-driven. The channel/slot utilization depends on the efficiency of the packet scheduling algorithm adopted by the master. The scheduling resources are slots of 625μs in length and each corresponds to a pseudo-random hopping frequency. To maximize the piconet performance, various intra-piconet scheduling algorithms have been designed, please refer to [2,6] and the references therein for details.

On the other hand, data exchange in a scatternet is the coordination result of both *intra-piconet* scheduling and *inter-piconet* scheduling [3,5]. Inter-piconet scheduling determines the schedule that a gateway should follow in relaying packets among different directly connected/neighboring piconets. A gateway can be present in at most one connected piconet at a time due to the multiple channel frequency hopping characteristic of the Bluetooth. So a gateway tends to become a traffic hotspot and congestion can be easily developed there.

We use scatternet scheduling to denote the combined effort of intra-piconet scheduling and inter-piconet scheduling. The key issue in designing scatternet scheduling algorithm is to determine the gateway's rendezvous points (RPs) in all directly connected piconets. A *rendezvous point* is a well-defined time point at which the gateway will switch to a certain connected piconet. In Fig. 1, gateway 0 needs to schedule its RPs in the three connected piconets, controlled by master nodes 2, 3 and 6 respectively. The amount of time required to visit all connected piconets (at least) once forms a *superframe* schedule. (Fig. 2 shows several superframe designs for the scatternet in Fig. 1.)

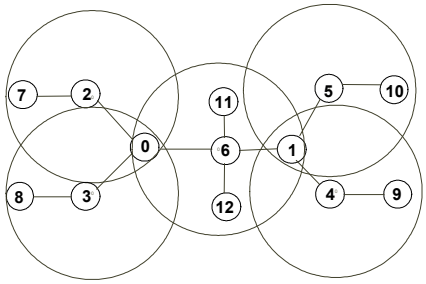

Fig. 1: An example scatternet.

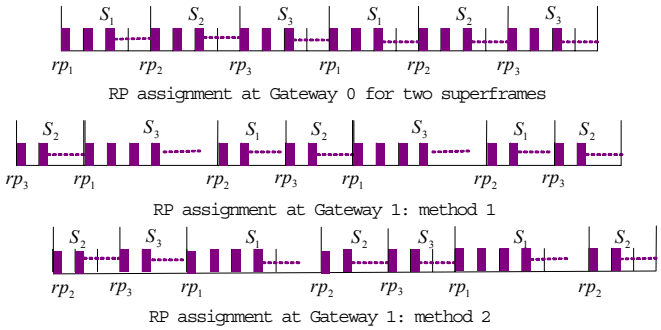

Fig. 2: Superframes obtained using MDRP for the scatternet in Fig.1. (S_i: the gateway's sojourn time in piconet i. rp_i: the time at which a gateway visits piconet i.)

II. EXISTING WORK

In [3], P. Johansson *et al* proposed a static scatternet scheduling algorithm called Maximum Distance Rendezvous Point (MDRP). MDRP uses the notion of a periodic *superframe* that is known to *all* nodes involved in the scatternet. To establish a new RP (rendezvous point) between a gateway (that already has i established RPs with other piconets/masters) and a master (that already has j established RPs with other gateways), the master has to map these $i+j$ established RPs onto a common superframe and choose the new RP as the middle point of the largest interval between any two successive RPs in the common superframe.

MDRP is a simple and efficient scheme, but it cannot adapt to the changes in traffic load as its gateway schedule (i.e. its superframe structure) is static. Besides, the amount of time (in each superframe) that a gateway stays with each connected piconet is in general not the same. This is due to MDRP's progressive manner in schedule

0-7803-7954-3/03/$17.00 ©2003 IEEE.

calculation - an inherent weakness in its design. Consider the example shown in Fig. 2, where the superframes are obtained using MDRP for the scatternet in Fig. 1. Note that there are various ways to assign RPs in MDRP depending on its configuration and the order of RP assignment [3] (also refer to Section IV. C). It can be proved that, with regard to Fig. 1, none of the possible schemes in MDRP can achieve equal splitting of the superframe. From the results of two common RP assignment schemes for Gateway 1 in Fig. 2, we can see that Gateway 1 spends 1/2 of the superframe time in one piconet and just 1/4 in the other two piconets. The incurred adverse effect due to *unequal* sojourn time splitting will be evaluated via simulations in Section IV.

As the traffic load varies, a static superframe design cannot efficiently utilize the available bandwidth. In [4], special fields in the packet payload are defined to piggyback queue length information between the communication peers; based on this feedback information, the RPs in a superframe can be tuned. However, the additional payload modification requirement makes it impractical for specification-compliant implementation. In [5], instead of calculating the *strict* local communication schedules, the rendezvous point is defined as the sniff slot [1] at which the inter-piconet communication *may start*. Lacking mutual schedule coordination, slots will be wasted if the gateway-master pair does not sniff (i.e. visiting a rendezvous point) at the same time. To alleviate this problem, an assumption has to be made that each device is able to quickly determine whether the peer device is active in the same piconet or not. This acts against the operation of sniff mode in Bluetooth Spec. [1], which requires a device to sniff for at least a certain pre-defined number of time slots. Further, work in [3,4] assumes the gateway can only be a common slave; this restricts their applicability in some scenarios where a master/slave gateway is preferable.

III. GATEWAY ORIENTED SCATTERNET SCHEDULING (GOSS) ALGORITHM

In this section, a simple scatternet-scheduling algorithm, called GOSS (gateway-oriented scatternet scheduling), is proposed. GOSS operates at the gateway. Unlike MDRP, establishing new RPs for a gateway does not need to consider the already established RPs. The only information needed is the number of piconets that a gateway should be directly connected to (so RPs with them should be established simultaneously). Initially, the superframe of a gateway, with a given initial size, is equally[1] divided among all directly connected piconets. As data exchange takes place, traffic statistics can be collected to fine-tune the initial superframe design (such that the gateway will stay longer in the piconet with more traffic).

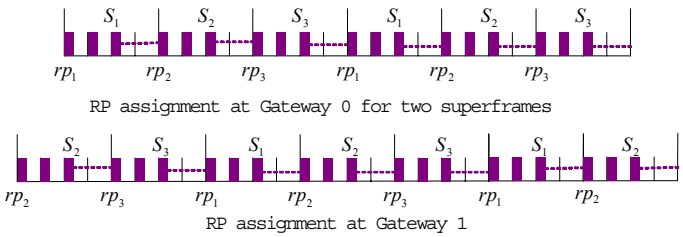

Fig. 3: RP assignment for scatternet in Fig. 1 using GOSS

[1] This is applicable to the gateway being a common slave, while for the master/slave gateway a different partition strategy may be needed. Due to space limitation, the details can be found in [7].

A. Static GOSS

Fig. 3 shows the initial superframes designed for the scatternet in Fig. 1 using GOSS algorithm. Note that the initial RP assignment in GOSS is gateway-oriented and operates in a real *static* manner, whereas MDRP has to work progressively even in a totally static scatternet, i.e., when no nodes join/leave. Compared with MDRP, we can show that RP assignment in GOSS is simpler, yet more efficient and flexible.

(1) Supporting arbitrary topology in GOSS

GOSS algorithm can support arbitrary topology, with no limitation on the role of the gateway posed, i.e., the gateway can be a common slave gateway or a master/slave gateway. For any scatternet topology that satisfies the condition that no more than two gateways directly connected with each other (e.g. the scatternet shown in Fig. 5), we can prove [7] that there always exists a RP assignment such that when a gateway enters a new piconet, its new "master" would remain active throughout its stay period.

(2) Adapting the superframe design to topology

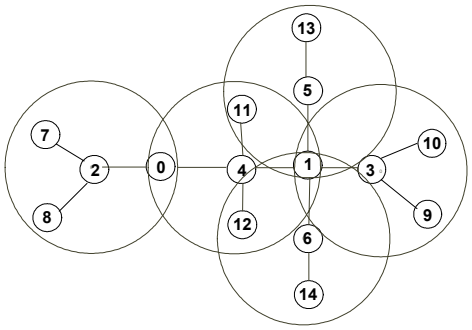

Fig. 4: a scatternet with two gateways of variable degrees.

In MDRP, the size of the superframe used by all the gateways is the same. We call it a *common superframe* scheme. In GOSS, since each gateway determines its own superframe independently, this flexibility endows us with a new strategy to calculate the superframe size based on the *degree* of the corresponding gateway, i.e. the number of neighboring piconets the gateway directly connected to. Fig. 4 shows a scatternet connected by two gateways (nodes 0 & 1), with degrees 2 and 4 respectively. By fixing the stay time at each piconet to be the same, the superframe size grows as the degree of a gateway. We call this *degree-based superframe* scheme.

The choice of common superframe or degree-based superframe is traffic dependent. In general, we prefer the degree-based superframe arising from the following observation and thinking. Being a gateway connecting to multiple piconets, it functions more like a relay node rather than a traffic source or sink. As a result, most of time the gateway bordering with multiple piconets will fetch data from one master and relay to another. This visualized granularity of operation prefers equal amount of gateway sojourn time in each neighboring piconet. Nevertheless, further adapting the superframe design to the traffic fluctuations can give additional performance improvement. We shall address this towards the end of this section.

(3) Adapting the superframe design to traffic demand

In GOSS, we also take the following practical constraints into considerations:

- *Switching overhead:* A gateway cannot immediately synchronize with the master in the piconet it newly joined.

This is because a master can only initiate a polling at even time slots, the gateway has to wait until the next even slot arrives. The time required for the gateway to synchronize its own clock with the new master is upper-bounded by two time slots (625 μs each). Hence frequent switching among connected piconets is not desirable.

- *Bandwidth wasted due to switching:* Bluetooth supports three packet sizes, 1, 3, or 5 time slots. Due to Bluetooth's duplex transmission, the lower bound of the stay period in each piconet should be 6 time slots, which corresponds to the case that a 5-slot packet in one direction, and a 1-slot packet/ACK in another. When a gateway is to be switched to the next piconet, the current head of line packet, associated with the current piconet, would probably be delayed until the next RP since it may not be able to complete its transmission in time. The smaller superframe size is, the larger percentage of possible bandwidth wastage would be.

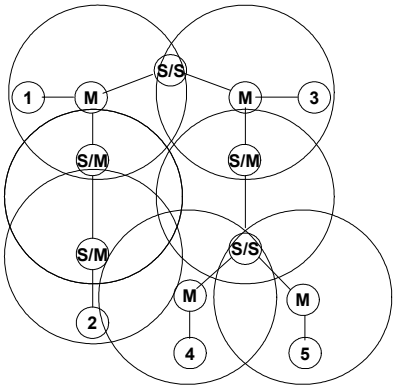

Fig. 5: A complex scatternet topology combining various basic components.

From the description above, we can see that instead of choosing the superframe as small as possible, we should strike a balance among the concomitant advantages using smaller superframe, the possible switching overhead, bandwidth wastage, etc. In Section IV. B, we use simulations to study the effect of varying traffic load on the scatternet scheduling performance with different superframe sizes.

B. Adaptive GOSS

We also propose a simple adaptation scheme which can dynamically adjust the superframe size based on the time-varying traffic load. Assume the gateway has n neighboring piconets. The stay period in each piconet S_i varies according to the link utilization as shown in Fig. 6. Here we use the link utilization as the indicator of the amount of traffic currently burdening the gateway. S_i is initially set to and remained at l until the corresponding link utilization exceeds α; after which S_i increases linearly m time slots each step. On the other hand, if the link utilization falls below α, S_i decreases at the same rate of m time slots per step, until l is reached. Note that the stay period S_i is upper-bounded by u and lower-bounded by l, where l and u are user defined.

If the link utilization becomes 0 when $S_i = l$, S_i would directly drop to 6. This allow us to utilize the gateway as efficiently as possible, by taking advantages of the fact that some neighboring piconets have no packets to send or receive, and at the same time to reserve the capability to detect any newly arrived backlogged packets as early as possible (so S_i does not drop to 0 here).

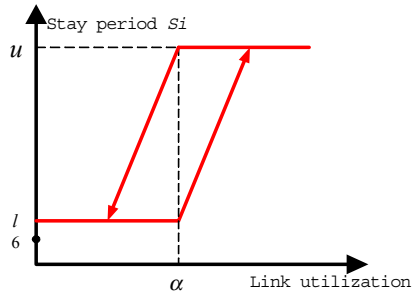

Fig. 6: adapting stay period based on link utilization.

Another important issue is how frequent should each gateway adjusts its frame size. Continual adaptation may enable the gateway follow the traffic change closely, but too frequent adjustments may burden the gateway with heavy signaling overheads (due to negotiation of the new sniff parameter). Worse still, it may cause fluctuation of the superframe size, as the link utilization fluctuates.

IV. PERFORMANCE EVALUATION

A. Simulation model

In this section, we compare the performance of GOSS with MDRP and give some guidelines in choosing the parameters for GOSS. For fair comparison, the same intra-piconet scheduling algorithm, Round Robin [6], is used in both MDRP and our GOSS. Like MDRP, each master endues priority to the gateway as each gateway only spends a fraction of time in one piconet. The gateway is exhaustively polled in the sense that as long as it has packets to send or receive the master persists polling it. Note that in MDRP only one gateway can be active at a time. To save the efforts/delay of global synchronization among gateways' schedules, GOSS allows multiple gateways to participate in a piconet simultaneously by alternating their transmission in a round robin manner.

Four different representative scatternet topologies are simulated to demonstrate various aspects of GOSS. To be more specific, the scatternet in Fig. 7 is used in Section IV.B for obtaining some guidelines in choosing superframe size. The scatternet in Fig. 1 is used in Section IV.C for studying the adverse effect due to unequal sojourn time splitting. The scatternet in Fig. 4 represents the scenario of having gateways of variable degrees. It is examined in Section IV. D. Finally in Section IV.E, we test our proposed traffic adaptation techniques based on the scatternet in Fig. 5, a complex all-inclusive topology.

Assume packets arrive at each queue in bursts[2] following a Poisson process with λ packet bursts/ms. Let the burst size be geometrically distributed with a mean of 4 packets. By varying λ, we can vary the total system load, and thus alter the system throughput. The buffer size at both master and slave is set large enough to preclude buffer overflow. Each point of simulation data is collected by simulating 50000 time slots with the initial 1000-slot statistics ignored.

The following performance metrics are used in our performance evaluations: 1) *Average packet end-to-end delay* (including both the

[2] For example, an IP datagram will be segmented into a burst of L2CAP packets for transmission over scatternet. Random segmentation among the packet size of 1, 3, 5 time slots are used in our experiment.

queuing delay at each hop and the transmission delay). 2) *System throughput* (the total number of packets successfully delivered per millisecond). 3) *Switching overhead* (percentage of the bandwidth wasted due to switching).

B. Choosing static superframe for GOSS

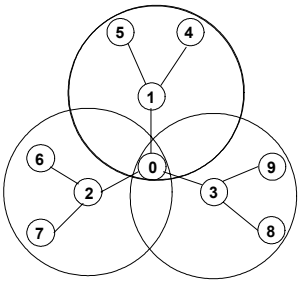

Fig. 7: scatternet consisting of three piconets

Fig. 8 studies the effect of varying superframe size in GOSS based on Fig. 7, a simple scatternet topology with one gateway shared by three piconets. We consider 12 unidirectional traffic flows, with the following (source, destination) pairs: (4,6), (4,7), (5,6), (5,7), (6,4), (6,5), (7,4), (7,5), (8,6), (8,7), (9,4), and (9,5). The same traffic rate λ is applied at each source node. Among all the choices considered, the superframe sizes of 45 and 150 perform the best in case of light and heavy traffic loads respectively. The singularity can be observed for the curves with frame sizes of 24 and 30, whose performance deteriorates dramatically regardless of the traffic load. The reason can be deduced from their serious bandwidth wastage and switching overhead in using too small frame sizes.

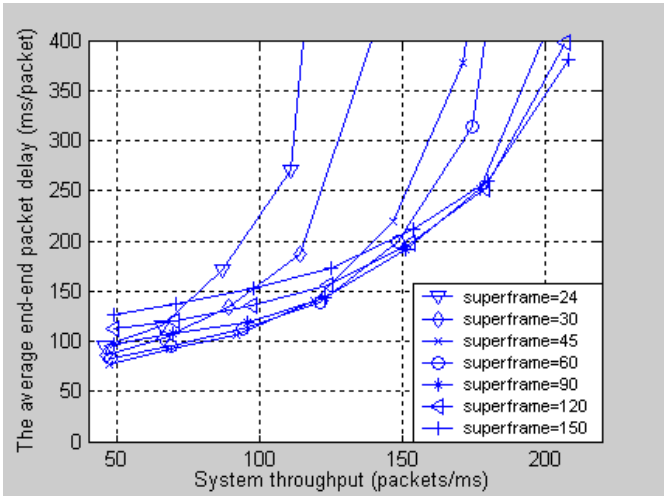

Fig. 8: performance comparison of different superframe sizes in GOSS

The switching overhead has been considered in our implementation. It can be approximated by the simple formula 1*gateway_degree/(superframe_size+1*gateway_degree), as each time the gateway switches, on average one time slot (worst case two time slots) will be wasted for synchronization. For example, in Fig. 7 the degree of the gateway is 3, if the superframe size is 30, the switching overhead is about 9%; while for a superframe size of 60, it decreases to 4.7%.

C. Comparing static GOSS with MDRP

Based on the topology of Fig. 1, Figs. 9 & 10 compare the static GOSS with two RP assignment schemes using MDRP (depicted in Fig. 2). For fair comparison, the superframe size is fixed to 120 for both MDRP and GOSS. Fig. 9 is obtained with equal *bidirectional* traffic flows between node pairs (8,9), (7,10), and (11,12), while Fig. 10 considers three *unidirectional* traffic flows between the same set of node pairs. Referred to Fig. 2, MDRP derives its gateway schedule in the following progressive manner. First, Gateway 0 calculates its own schedule by assigning its stay periods with masters 2, 4, and 6 exactly 1/3 of its superframe size. Then depending on the order of establishing the switching points for Gateway 1, two different assignment schemes are possible. If Gateway 1 establishes its schedule with masters 4 and 5 first, then its stay period with master 6 will be 1/4 of the superframe size, which corresponds to method 2 in Fig. 2 (denoted as MDRP2); otherwise, its stay period with master 6 will be 1/2 of the superframe size, which corresponds to method 1 in Fig. 2 (denoted as MDRP1). From Figs. 9 & 10, we can see that GOSS outperforms both versions of MDRP. Besides, we observe a large performance gap between MDRP1 and MDRP2. As such some guidelines or heuristics are needed in MDRP algorithm for selecting among multiple RP assignment choices. But for our GOSS, it inherently avoids such complexities.

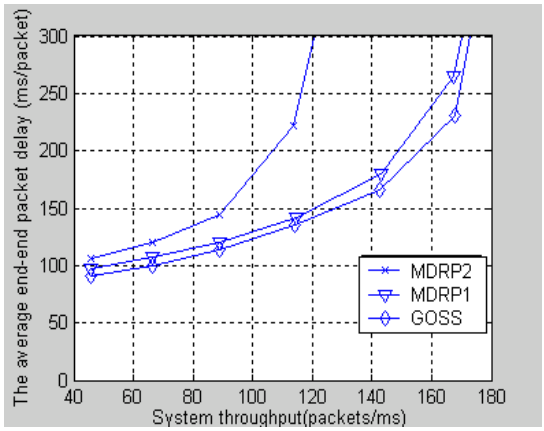

Fig. 9: performance comparison of GOSS and MDRP (bidirectional traffic)

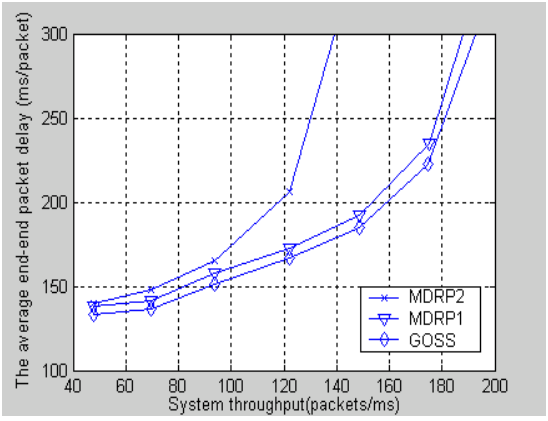

Fig. 10: performance comparison of GOSS and MDRP (unidirectional traffic)

D. Degree based superframe VS. common superframe

Based on the scatternet in Fig. 4, Figs. 11 & 12 study the degree-based superframe size design versus the common superframe size design. Assume a gateway should stay with each connected piconet for 40 time slots. Using the degree-based scheme, the

superframe sizes of Gateway 0 and Gateway 1 are 80 (=40x2) and 160 (=40x4) respectively. With common superframe size design, both gateways will use the same frame size of 160.

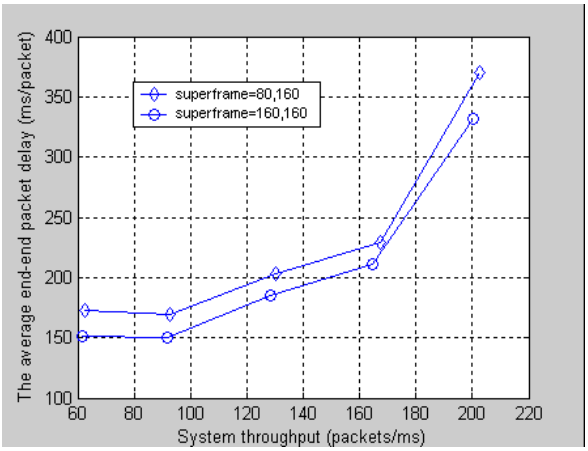

Fig. 11: degree-based superframe against common superframe (bidirectional traffic)

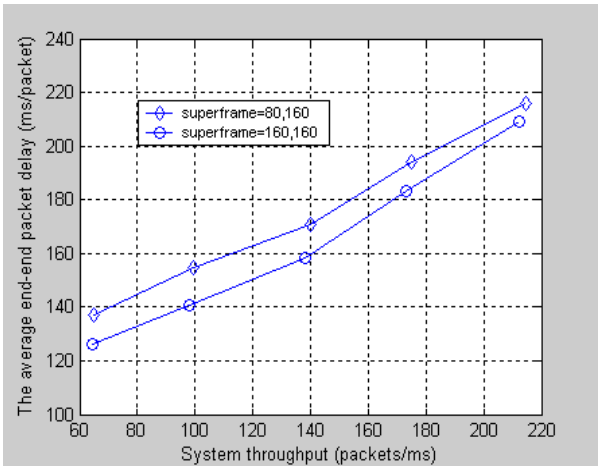

Fig. 12: degree-based superframe against common superframe (unidirectional traffic)

Fig. 11 is obtained with equal bidirectional traffic flows between node pairs (8,9), (7,10), (11,12), and (13,14), whereas Fig. 12 is based on 4 unidirectional traffic flows between the same set of node pairs. It happens that for the scatternet shown in Fig. 4, both MDRP and GOSS give the same RP assignment. Therefore all the performance improvement shown in Figs. 11 & 12 is brought by using the degree-based superframe size design.

E. Adapting superframe to the traffic

Fig. 13 compares the performance of adaptive GOSS (with l=15, u=50) and static GOSS (with two different stay periods: 15 & 50 time slots in each connected piconet) based on the scatternet shown in Fig. 5. The adaptation technique used here follows the linear curve in Fig. 6, the step size for sojourn time adjustment is 5 time slots. Parameter α is set to 0.6 with the adapting frequency of every 2 superframes. Note that the link utilization is monitored constantly by maintaining two counters, one for used time slots and one for idle time slots. The incurred signaling overhead would be around 1/(2*superframe_size), since at least one slot is needed each time the gateway conveys the new sniff parameters towards the corresponding master. Four bidirectional traffic flows along the shortest path between node pairs (1,2), (1,3), (3,4), and (3,5) are simulated.

As expected, from Fig. 13 the adaptive scheme is a clear winner irrespective of traffic load. Besides, unlike MDRP which is sensitive to the initial RP assignment at each gateway, GOSS algorithm can remove the undesirable "synchronization" possibly posed by the initial configuration with the periodic adjustment of the superframe design. This leads to the substantial gains in system performance in Fig. 13.

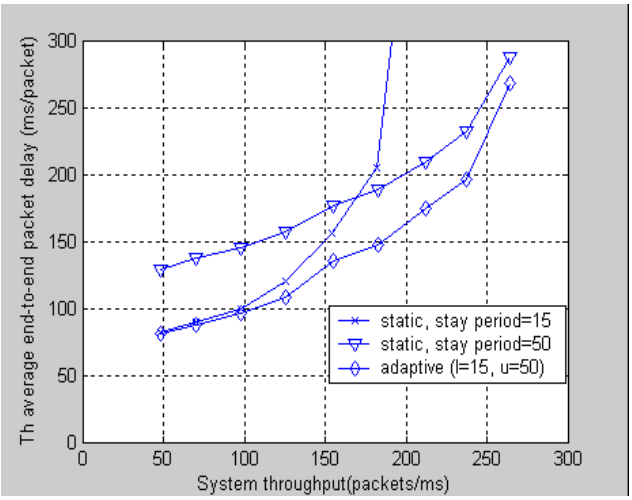

Fig. 13: adaptive GOSS versus static GOSS

V. CONCLUSIONS

A simple adaptive MAC scheduling algorithm, called GOSS, is proposed for Bluetooth scatternet. To avoid the progressive calculation of the switch schedule in a static environment, GOSS is designed to be gateway-oriented in order to ensure equal partition of the superframe at each gateway. At the same time, extension can be easily made to support arbitrary scatternet topology carrying dynamic traffic loads, where nodes can join/leave at any time. Simulation results showed that even a static GOSS prevails over the existing scheme. If the adaptation technique is used, further performance enhancement can be obtained.

REFERENCES

[1] Bluetooth Special Interest Group, "Specification of the Bluetooth System 1.1," http: //www.bluetooth.com/

[2] A. Das, A. Ghose, A. Razdan, H. Saran and R. Shorey, **"**Enhancing performance of asynchronous data traffic over the Bluetooth wireless ad-hoc network," *IEEE INFOCOM 2001*, Anchorage, Alaska, April 2001

[3] P. Johansson, R. Kapoor, M. Kazantzidis, M.. Gerla "Rendezvous scheduling in bluetooth scatternets," *IEEE International Conference on Communications, ICC 2002*, Volume 1, pp. 318 –324

[4] Wensheng Zhang and Guohong Cao "A flexible scatternet-wide scheduling algorithm for Bluetooth networks,**"** *21st IEEE International Conference on Performance, Computing, and Communications*, 2002, pp. 291 –298.

[5] S. Baatz,. M. Frank, C. Kuhl, P. Martini, C. Scholz, "Bluetooth scatternets: An enhanced adaptive scheduling scheme," *INFOCOM 2002.*

[6] Changlei Liu, Kwan L. Yeung and Victor O.K. Li, "A Novel MAC Scheduling Algorithm for Bluetooth System," *GLOBECOM 2003*, in press.

[7] Changlei Liu, "Bluetooth Network Design," MPhil. thesis, the University of Hong Kong, 2003