Mobile Networks and Applications 9, 9–20, 2004

A Fair and Traffic Dependent Scheduling Algorithm for Bluetooth Scatternets

ROHIT KAPOOR
University of California, Los Angeles (UCLA), USA

ANDREA ZANELLA
University of Padova, Italy

MARIO GERLA
University of California, Los Angeles (UCLA), USA

Abstract. The Bluetooth specification defines the notion of interconnected piconets, called scatternets, but does not define the actual mechanisms and algorithms necessary to set up and maintain them. The operation of a scatternet requires some Bluetooth units to be inter-piconet units (gateways), which need to time-division multiplex their presence among their piconets. This requires a scatternet-scheduling algorithm that can schedule the presence of these units in an efficient manner. In this paper, we propose a distributed scatternet-scheduling scheme that is implemented using the HOLD mode of Bluetooth and adapts to non-uniform and changing traffic. Another attribute of the scheme is that it results in fair allocation of bandwidth to each Bluetooth unit. This scheme provides an integrated solution for both intra- and inter-piconet scheduling, i.e., for polling of slaves and scheduling of gateways.

Keywords: Bluetooth, scatternet, scheduling, fairness

1. Introduction

The Bluetooth [10] technology was developed as a replacement of cables between electronic devices and this is perhaps its most obvious use. But, it is the ability of Bluetooth devices to form small networks called piconets that opens up a whole new arena for applications where information may be exchanged seamlessly among the devices in the piconet. Typically, such a network, referred to as a PAN (Personal Area Network), consists of a mobile phone, laptop, palmtop, headset, and other electronic devices that a person carries around in his every day life. The PAN may, from time to time, also include devices that are not carried along with the user, e.g., an access point for Internet access or sensors located in a room. Moreover, devices from other PANs can also be interconnected to enable sharing of information.

The networking capabilities of Bluetooth can be further enhanced by interconnecting piconets to form scatternets. This requires that some units be present in more than one piconet. These units, called gateways, need to time-division their presence among the piconets. An important issue with the gateways is that their presence in different piconets needs to be scheduled in an efficient manner. Moreover, since the gateway cannot receive information from more than one piconet at a time, there is a need to co-ordinate the presence of masters and gateways.

Some previous work has looked at scheduling in a piconet [2,5] and also in a scatternet. In [4], the authors define a Rendezvous-Point based architecture for scheduling in a scatternet, which results in the gateway spending a fixed fraction of its time in each piconet. Such a fixed time-division of the gateway may clearly be inefficient since traffic is dynamic. In [9], the authors propose the Pseudo-Random Coordinated Scatternet Scheduling (PCSS) scheme in which Bluetooth nodes assign meeting points with their peers. The sequence of meeting points follows a pseudo-random process that leads to unique meeting points for different peers of a node. The intensity of these meeting points may be increased or decreased according to the traffic intensity. This work presents performance results for various cases. In [11], a scatternet-scheduling algorithm based on the concept of a switch table, which can be dynamically adjusted based on traffic load, is presented. In [1], the authors present a credit-based scheduling scheme based on the SNIFF mode of Bluetooth, where credits may be reallocated to cater to changing traffic.

Our scheduling scheme addresses the issues of fairness and utilization of bandwidth. Since Bluetooth is a low-bandwidth environment, it is important that bandwidth should be efficiently utilized. Also, since a low bandwidth can easily lead to starvation of flows, another metric we focus on is fairness. We propose a distributed scatternet-scheduling algorithm that is implemented using the HOLD mode [10] of Bluetooth and adapts to non-uniform and changing traffic. This algorithm provides an integrated solution for both intra- and inter-piconet scheduling, i.e., for polling of slaves and scheduling of gateways. The algorithm leads to a high bandwidth utilization and results in a fair division of (a) the piconet bandwidth between the slaves of a piconet and (b) the gateway presence among different piconets.

In section 2, we discuss the Bluetooth technology. In section 3, we present a definition of fairness in the context

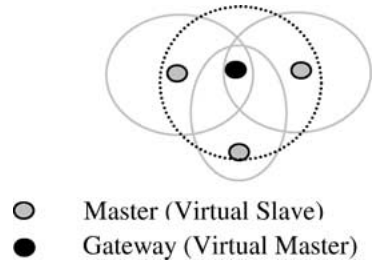

Figure 1. Gateway may be viewed as a virtual master and masters as virtual slaves.

of Bluetooth scatternets, which takes into account intra- and inter-piconet max-min fairness. Section 4 describes the algorithm and proves its fairness property. Section 5 presents simulation results and section 6 presents the conclusions.

2. Bluetooth technology

The Bluetooth system [3] operates in the worldwide unlicensed 2.4 GHz Industrial–Scientific–Medical (ISM) frequency band. To make the link robust to interference, it uses a Frequency Hopping (FH) technique with 79 radio carriers. It allows a raw data transmission rate of 1 Mbit/s.

Two or more Bluetooth units sharing the same channel form a piconet. Each piconet consists of a master unit and up to seven active slave units. The master unit polls the slave units according to a polling algorithm and a slave is only allowed to transmit after the master has polled it. The piconet capacity is thus, shared among the slave units according to the polling algorithm.

Furthermore, two or more piconets can be interconnected, forming a scatternet. This requires a unit, called an inter-piconet unit (gateway), to be a part of more than one piconet. Such a unit can simultaneously be a slave member of multiple piconets, but a master in only one, and can transmit and receive data in only one piconet at a time; so participation in multiple piconets has to be on a time-division multiplex basis. The time of the gateway is, thus, also shared among the piconets it belongs to. In this work, we assume that the gateway can only be a slave in its piconets. If a gateway were to be a master in a piconet, it would lead to the stoppage of all transmission in the piconet when the gateway visits some other piconet. Thus, we believe that the use of the gateway as a slave is the most efficient method of scatternetting.

3. Fair allocation of bandwidth

As introduced in the previous section, units belonging to a piconet share the piconet capacity according to the polling algorithm used by the master. In an analogous manner, gateways in a scatternet divide their time among their different piconets, according to the "master-listening" algorithm they use. It can be noted that there is a duality in this architecture. On the one hand, a master divides its capacity among the units of its piconet by using a polling algorithm. On the other hand, a gateway shares its capacity among the piconets it belongs to, on the basis of a scheduling algorithm it uses for listening to the masters. The gateway, can, then be viewed as a "virtual master" and its masters can be viewed as "virtual slaves" forming a "virtual piconet", in which the polling cycle is, actually, the "listening cycle" of the gateway. A graphical interpretation of this duality is given in figure 1, in which the solid line shows the actual piconets, and the dotted line shows the virtual piconet.

Due to this duality, we design our scheduling scheme such that the same scheduling algorithm is used for fair sharing of both (a) the piconet capacity among slaves and (b) the gateway time among piconets.

We now give a definition of max-min fairness [7]. We then go on to define max-min fairness in the context of Bluetooth scatternets, by considering (a) intra-piconet fairness, i.e., fairness in division of piconet bandwidth among slaves (both gateway and non-gateway) of a piconet and (b) inter-piconet fairness, i.e., fairness in division of the gateway's presence among its piconets. We first define a 'feasible' rate distribution since this is used in the definition of max-min fairness.

Definition 1 (Feasible). A rate distribution is *feasible* if rates are non-negative, the aggregate rate is not greater than one, and no unit receives a higher rate than required.

Definition 2 (Max-min fairness). An allocation of rates η_1, $\eta_2, \ldots, \eta_s$ among s units is *max-min fair* if it is feasible, and for each unit i, η_i cannot be increased (while maintaining feasibility) without decreasing η_j for some other unit j for which $\eta_j \leqslant \eta_i$.

The distribution of max-min fair rates depends upon the set of rate demands (traffic generated) of the units. In the following subsections, we discuss factors that determine the max-min "fair share" of a slave (gateway or non-gateway). We call these factors the *Piconet Presence Fraction* and the *Scatternet Presence Fraction* and show how they may be used to calculate the "fair share" for a slave in a scatternet.

3.1. Piconet presence fraction

Consider a piconet consisting of gateway and non-gateway slaves in which the master has complete knowledge of the rate demands of all slaves (an ideal master). Using this knowledge, the master polls the slaves in a max-min fair manner such that each slave gets its "fair share" of the master's polling. We refer to the "fair share" received by a slave as the "piconet presence fraction" (PPF) of the slave. The gateway has a PPF for each piconet it belongs to.

Consider the piconets shown in figures 2(a) and 2(b), each consisting of one gateway and two slaves, with the traffic rates of each slave as shown. In figure 2(a) (Piconet I), the PPF of each non-gateway slave is 0.2, while the PPF of the gateway is 0.6. In figure 2(b) (Piconet II), the PPFs of the slaves are 0.2 and 0.4, while the PPF of the gateway is 0.4.

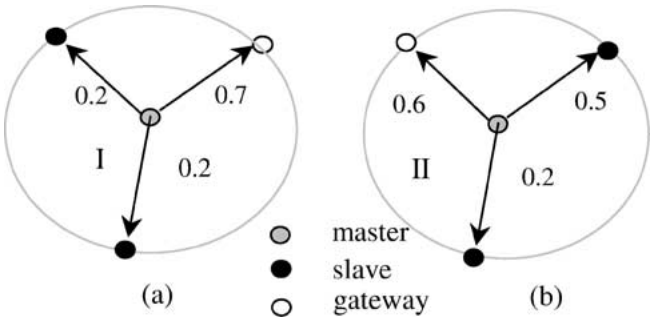

Figure 2. Piconets with traffic rates between master and each slave shownn.

3.2. Scatternet presence fraction

A gateway will, in general, be a slave in multiple piconets and may have different amounts of traffic to exchange with each piconet. Consider an ideal gateway that has complete knowledge of the rate demands of all its masters. The gateway can then divide its presence among its piconets in a max-min fair manner, giving each piconet a "fair share" of its presence. We call this fair share the "scatternet presence fraction" (SPF) of the gateway for the piconet. The importance of the SPF is that a fair division of the gateway's presence among its piconets can be achieved based on the SPF.

Consider the piconets of figure 2 again, but the gateway of each of the piconets now connects them to form a scatternet, as shown in figure 3. The traffic requirements are the same as shown in figure 2. The SPF of the gateway is 0.5 in Piconet I and 0.5 in Piconet II.

3.3. Fair share

We see that for a gateway to be fair, there are two kinds of fairness it has to achieve: that dictated by the PPFs, which achieves fairness between the gateway and the other slaves of a piconet, and that of the SPFs, which distributes the presence of the gateway between its piconets in a fair manner. Both these kinds of fairness may not always be completely achievable and this can lead to a change in the values of PPF and SPF, as we now discuss.

We observe that an ideal master (as in section 3.1) does not give a gateway more than the PPF of its polling. Thus, if the SPF of a gateway is greater than its PPF for a piconet, the gateway spends a fraction of its time equal to the PPF in the piconet. The gateway cannot stay for a fraction equal to its SPF in the piconet since it is limited by its PPF. Thus, the extra scatternet presence fraction (the difference of the SPF and the PPF) is redistributed in a fair manner among the gateway's other piconets for which the SPF is less than the PPF. This may increase the SPF of the gateway in the other piconets. In other words, the gateway behaves as if its SPF in a particular piconet is reduced to the PPF and thus, its SPF in the other piconets increases. We refer to this changed SPF as the "updated SPF" of the gateway in a piconet.

Similarly, an ideal gateway does not stay a fraction of time more than the SPF in a piconet. Thus, if the PPF of the gateway in the piconet is greater than the SPF, the gateway spends a fraction of time equal to the SPF in the piconet. The remaining PPF of the gateway (the difference of the PPF and the SPF) is redistributed in a fair manner among the other slaves of the piconet (if this other slave is a gateway, it is redistributed to it if its SPF is greater than its PPF in the piconet). This may increase the PPF of these slaves. We refer to this changed PPF as the "updated PPF" of the slave in the piconet. In case there is no such redistribution, the updated PPF is equal to the PPF and the updated SPF is equal to the SPF.

Table 1
Calculation of fair share of the gateway in the two piconets of figure 3.

	Piconet I	Piconet II
Actual traffic rate	0.7	0.6
PPF	0.6	0.4
SPF	0.5	0.5
Updated PPF	0.6	0.4
Updated SPF	0.6	0.4
Fair share	0.6	0.4

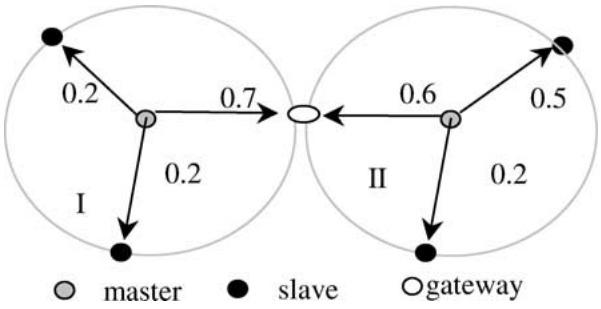

Figure 3. Gateway shared between two piconets; traffic rates between slaves and the master are shown.

The fair share can now be calculated from the "updated PPF" and the "updated SPF" as the minimum of these two quantities. Note that all these quantities – PPF, SPF, updated PPF, updated SPF and fair share–are dependent on the traffic. Any change in traffic demand of a unit may lead to a change in some of these quantities. We explain the calculation of the fair share using some examples.

An example is given in table 1, which shows the actual traffic rate, PPF, SPF, Updated PPF, Updated SPF and fair share of the gateway in the two piconets of figure 3. In Piconet II, the gateway has a PPF of 0.4, which is less than the SPF. In Piconet I, the gateway has a PPF of 0.6 and an SPF of 0.5. Thus, the extra scatternet presence fraction of the gateway in Piconet II (the difference between the SPF and the PPF) is given to Piconet I, which has a higher traffic rate than may be allowed by the SPF. This is reflected in the "updated SPF" values. Thus, the "fair share" of the gateway in Piconet I is 0.6 and in Piconet II is 0.4. The fair shares of the non-gateway slaves are equal to their PPF.

As another example, consider the scatternet consisting of 5 piconets with the traffic rates shown as in figure 4. As shown in table 2, gateway G2 has a PPF of 0.5 and an SPF of 0.4 in Piconet B. Thus, the "updated PPF" of G2 in Piconet B is 0.4. The extra PPF ($=$ PPF $-$ SPF) is added to the PPF of gateway

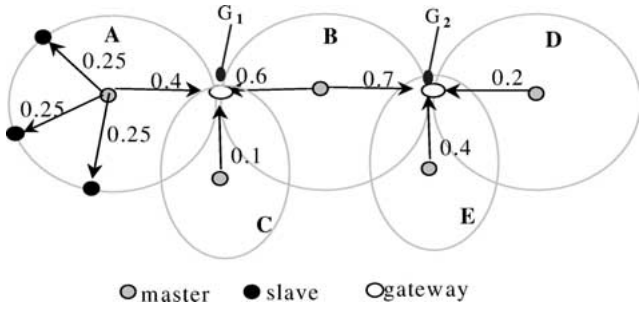

Figure 4. Scatternet with two gateways.

Table 2
Calculation of fair share of the gateways G1 and G2 in the scatternet of figure 4.

Gateway G1	Piconet A	Piconet B	Piconet C
Actual traffic rate	0.4	0.6	0.1
PPF	0.25	0.5	0.1
SPF	0.4	0.5	0.1
Updated PPF	0.25	0.6	0.1
Updated SPF	0.25	0.65	0.1
Fair share	0.25	0.6	0.1
Gateway G2	**Piconet B**	**Piconet D**	**Piconet E**
Actual traffic rate	0.7	0.2	0.4
PPF	0.5	0.2	0.4
SPF	0.4	0.2	0.4
Updated PPF	0.4	0.2	0.4
Updated SPF	0.4	0.2	0.4
Fair share	0.4	0.2	0.4

G1 in Piconet B. The "updated PPF" of G1 in Piconet B is, thus, 0.6.

Also, gateway G1 has a PPF of 0.25 and an SPF of 0.4 in Piconet A. Thus, the "updated SPF" of G1 in Piconet A is 0.25. The extra SPF (= SPF − PPF) is added to the SPF of G1 in Piconet B. The "updated SPF" of G1 in Piconet B, is thus, equal to 0.65. The fair shares can now be easily calculated.

A division of the master's polling and the gateway's presence based on PPF and SPF as described in this section takes into account the traffic demands of the slaves and the gateways and leads to fairness in the scatternet. In the next section, we introduce and describe an algorithm that aims to achieve such a fair distribution of bandwidth.

4. Description of algorithm

We first explain how the algorithm works in the case of a single piconet with no gateway. We then extend the algorithm to the case of a scatternet and explain how the coordination between the master and the gateways is achieved. We then prove the fairness of the algorithm.

4.1. Single piconet with no gateways

The polling algorithm is based on the master estimating the traffic rate between each slave and itself. This traffic rate is the sum of the traffic rates from the master to a slave and in the reverse direction. We assume, in order to simplify the explanation of the algorithm, that traffic flows only from slaves to master; masters generate no traffic to slaves. The same algorithm also applies with little change when traffic flows in both directions (explained later).

The master uses a Round Robin polling scheme, with the modification that a slave is skipped if it does not belong to the "active list" of the master. The slaves are moved in and out of the active list on the basis of two variables that the master maintains for each slave. These two variables are:

r – estimate of the rate of traffic generated by the slave;

N – estimate of the queue length of the slave.

When a slave is polled, the master–slave pair gets a chance to exchange a maximum amount of data in each direction, denoted by M. After each such polling phase, the master updates the values of N and r in the following manner:

For the slave just polled:

$$N = N + r\tau - x, \tag{1}$$

$$r = \begin{cases} \alpha r + (1-\alpha)\dfrac{x}{T}, & x < M, \\ \alpha r + (1-\alpha)\dfrac{x}{T} + \delta, & x = M. \end{cases} \tag{2}$$

For other slaves:

$$N = N + r\tau, \tag{3}$$

where τ is the time elapsed since the last update, x is the amount of data exchanged during the poll phase, T is the total time elapsed since the last poll of the same slave, α is a parameter used to smooth the rate estimation and δ is a parameter used to probe for more bandwidth. Note that x is the actual amount of data exchanged, which may be less than or equal to M, depending upon the number of packets in the slave's queue. Since N is an estimate of the slave's queue length and r is an estimate of the rate at which traffic is generated, N is increased at the rate of r (as in equations (1) and (3)). Also, when a slave is polled, N is decreased by the amount of data exchanged ((equation 1)).

After updating these values, the master determines the changes to be made to the active list. A slave is added or deleted from the active list depending upon whether its value of N is greater or smaller than a "threshold". The value of this threshold is the minimum amount of data that the master would like the slave to have in order to poll it. We choose a value equal to a multiple of a DH5 packet for the threshold since this packet incurs least overhead (the selection of the value of the threshold is discussed further in the next subsection). Thus, a slave is present in the active list if the master's estimate of the value of N for the slave is greater than the threshold. This makes the simple Round Robin polling strategy adaptive to traffic and enables it to utilize bandwidth efficiently, even when slaves have different rates of traffic. The maximum amount of data that can be exchanged at each poll, M, is also set equal to the threshold. Note that if the amount of data, x, in the slave's queue is less than the threshold, the polling of the slave ends after this data has been exchanged.

If the value of N is less than the threshold for all the slaves, then the slave whose value of N is estimated to take the smallest time to reach the threshold is polled, i.e., the slave for which the value of $(\text{Threshold} - N)/r$ is the smallest.

The master now goes to the next slave according to the Round Robin ordering of slaves. If the slave is present in the active list, it is polled. Else, the procedure is repeated for the next slave in the Round Robin ordering.

Also, note that if the amount of data sent by the slave x is equal to M, r is increased by a small amount, δ. This is basically an attempt by the slave to probe for more bandwidth if it is able to send data at the present rate. The usefulness of this increase is evident in the proof of fairness in the next section. The value of δ chosen is 0.15 and that of α is 0.65. We also discuss the rationale behind choosing these values in the proof of fairness.

If traffic flows in both directions, i.e., from the slaves to the master and in the reverse direction, x is the average of the amount of data exchanged in the two directions, r refers to the average of the rate-estimations of the two directions and N refers to the average of the queue length estimates of the two directions. Also, if the number of packets in either direction is less than the threshold, the polling of the slave continues till in both directions, (a) there is no more data to send or (b) amount of data equal to the threshold has been exchanged.

The initial value of N is set to the threshold (to ensure that slaves get polled at the beginning) and that of r is set to 0.25 (as a reasonable value). Note that the algorithm converges to the fair share, but a careful selection of initial values makes the initial convergence faster.

Another advantage of such a scheme is that it may allow the master to go into a power-saving mode if it realizes that no slave has sufficient packets to send, i.e., if N is smaller than the threshold for all slaves. Though we do not explore this option in this paper, it may be useful since Bluetooth devices are expected to work in power-constrained environments.

To improve the algorithm, we add a heuristic to it. The maximum number of polling cycles that a slave is not polled is bounded. If a slave generates a large burst of data occasionally and then does not generate any data for a long time, the value of r for the slave may be very low. This may cause the value of N for the slave to be lower than the threshold for a long time. By limiting the maximum number of cycles missed by the slave, we make sure that such a behavior of the slave does not lead to its starvation. In the experiments, this value is taken to be equal to 5 cycles. We now explain how the above algorithm works in a scatternet.

4.2. Scatternet

Scheduling of gateways using Rendezvous Points. Before describing how the algorithm works in a scatternet, we briefly discuss the notion of Rendezvous Points (RPs) described in [4]. A RP is a slot at which a master and a gateway have agreed to meet, i.e., at this slot, the master will poll the gateway and the gateway will listen to the master. In [4], RPs are implemented using the SNIFF mode of Bluetooth, but we implement RPs using the HOLD mode [10]. In the HOLD mode, the slave does not have to listen to the master for a certain time period and may use this time to visit other piconets. Prior to entering the HOLD mode, the master and the slave agree on the time duration the slave remains in the HOLD mode. We implement our algorithm using RPs as described below.

The working of the algorithm in a scatternet is very similar to its operation in a piconet. The master continues to poll the non-gateway slaves in the same manner as described in the previous section with the modification that a gateway is polled at a Rendezvous Point. Each RP is a slot at which a particular gateway is polled and a master has different RPs for each of its gateways. These RPs are always unique (i.e., a master cannot have the same RP with more than one gateway). Since the gateway must be polled at the RP, this has implications in the polling of the other slaves (discussed later). Once a gateway has been polled, the master continues with the polling of the other slaves in the same manner as described in the previous section, i.e., it checks its active list to see if the next slave in the polling cycle is to be polled and so on.

In order to divide its time among different piconets in a fair manner, the gateway performs similar calculations as described in the earlier section for the master. The gateway maintains values of N and r for each piconet it belongs to and these values are updated each time a gateway is polled (i.e., at each RP). Thus, the calculations performed by a gateway at each RP are:

For the piconet in which the gateway was just polled:

$$N = N + r\tau - x, \tag{4}$$

$$r = \begin{cases} \alpha r + (1-\alpha)\dfrac{x}{T}, & x < M, \\ \alpha r + (1-\alpha)\dfrac{x}{T} + \delta, & x = M. \end{cases} \tag{5}$$

For other piconets:

$$N = N + r\tau, \tag{6}$$

where τ is the time elapsed since the last update, x is the amount of data exchanged during the poll phase, T is the total time elapsed since the gateway was polled in the same piconet, and α and δ are as defined earlier.

Moreover, at each RP, the gateway and the master negotiate the next RP between them. The assignment of this next RP takes into account the fairness between (a) the gateway and other slaves in a piconet and (b) the presence of the gateway in different piconets. Also, we again employ a heuristic that improves the algorithm. When the next RP is being negotiated, we keep a bound on the maximum value this can take. This prevents a piconet from not being visited by a gateway for a long time. The maximum value of this next RP used in our experiments is 400 slots.

We now see how the master and the gateway use the information that they have to achieve fairness in the scatternet. When a gateway is polled at a RP, the gateway and the master do the following.

(i) *Gateway.* The gateway calculates the number of slots, N_{thresh} after which N for the piconet will become greater than the threshold; $N_{\text{thresh}} = (\text{threshold} - N)/r$, where threshold is as explained in the previous section, N and r are values maintained by the gateway for the piconet. The gateway makes use of this value and does not visit a piconet till its estimate of N for the piconet becomes greater than the threshold. This is similar to the algorithm used by the master in which a slave is not polled till the master's estimate of N for the slave becomes greater than the threshold. Thus, the gateway tries to divide its time between the piconets in a fair manner, i.e., according to the SPFs. Note that N_{thresh} may be negative if N is greater than the threshold. Also, N_{thresh} is allowed to have a maximum value of 400.
Moreover, each time a gateway visits a piconet, it knows the RPs for the other piconets it belongs to (except right at the beginning or when the gateway is added to another piconet).

(ii) *Master.* The master calculates the number of slots after which the gateway can be polled such that the fairness with other slaves is maintained. It adopts the following procedure to achieve this:
It maintains a counter, *num_slots* (which is initialized to 0) and checks the value of N for each slave, in a cyclic order, starting from the slave after the current gateway in the cyclic order to the slave before the current gateway. The master checks if the value of N for the slave will be greater than the threshold after *num_slots* slots. If this condition is true, *num_slots* is incremented by twice the value of the threshold. After incrementing *num_slots*, the master also checks to see if it has a RP with any gateway whose value is equal to *num_slots* and increments *num_slots* by twice the value of the threshold if this is true. This ensures that the master has a unique RP for each of its gateways. Note that *num_slots* is incremented by twice the value of the threshold since the master expects to exchange threshold slots of data with a slave in each direction.
The master uses the above procedure to estimate the number of slaves who will have their value of N greater than the threshold when the master polls the slaves in their cyclic order starting from the gateway just polled. The value of *num_slots* determines the number of slots which the master expects to use in polling the other slaves in one cycle before polling the gateway again and is thus, used by the master to maintain fairness between the gateway and the other slaves in the piconet. Again, note that *num_slots* is allowed to have a maximum value of 400.

The master and the gateway now exchange the information they have to calculate their next RP. This exchange takes place using the LMP_hold_req PDU of the LMP (Link Manager Protocol) layer. This PDU carries a *hold instant* and a *hold time*, which are used to specify the instant at which the hold will become effective and the hold time, respectively. When the master is sending a packet to a gateway, the value of *num_slots* can be sent after *hold instant* and *hold time* in the packet. The master also sends the values of its RPs with its other gateways in the packet. Similarly, the gateway sends the master the values of its RPs with other piconets and the value of N_{thresh} also in an LMP_hold_req PDU. The master now knows all the RPs of the gateway; similarly, the gateway knows all the RPs of the master.

Note that the above information exchange requires a minimal change in the Bluetooth specifications that the contents of the LMP_hold_req PDU need to be enhanced. This PDU is 1-slot in length; thus, some bandwidth of the master is wasted in sending these PDUs. This wasted bandwidth can be reduced by increasing the value of threshold, i.e., the maximum data that a slave and a master may exchange in each direction during one poll of the slave. On the other hand, a large value of the threshold will lead to larger delays for packets. Thus, we have a tradeoff here. We choose a threshold value equal to three times a DH5 packet. The effect of this wasted bandwidth can be seen in the experiments section where the piconet capacity used is slightly less than 1. Note that we pay a small price here to get perfect coordination between the master and the gateway and also to get a high degree of fairness in the system, as the experiments later demonstrate.

Now, the master and the gateway both have complete information. So, each of them calculates the next RP in the following manner:

They take the maximum value out of *num_slots* and N_{thresh} and as long as this value is the same as one of the RPs (note that all relevant RPs are known to both the master and the gateway), the value is incremented by $2 \cdot \text{threshold}$. The value at the end of this small procedure is the next RP between the gateway and the master. Since this value takes into account both N_{thresh} and *num_slots*, it incorporates both the fairness of the master's polling and the gateway's presence.

Note that the value of *num_slots* calculated by the master is just an estimate (the master assumes that each slave included in the calculation of *num_slots* will exchange threshold slots of data with the master in each direction, but this may not be true). Thus, the master may have polled all the slaves that had to be polled before the RP of the gateway (according to the estimate in the calculation of *num_slots*) and still be left with some slots before the RP. In this case, the master just continues polling the slaves in their cyclic order and polls the gateway when the time for the RP arrives. Note that this means that the master may have to force a slave to send a packet smaller than a certain length. For example, if two slots are left for the RP, then the master will send a 1-slot packet and ask the slave being polled to do the same. Note that the Bluetooth header has 4 bits to represent the packet type and these can represent 16 packet types. For ACL links, 10 (7 data, 3 control packets) of the packet types are defined. We use 2 of the remaining bit sequences to send packets that force the slave to send packets smaller than or equal to a certain length. This is shown in table 3.

From table 3, we see that this procedure is adopted if the number of slots left for the RP is less than 10 (if the number of slots left for the RP is greater than or equal to 10, then the

Table 3
Procedure adopted by the master if slots left for the RP is less than 10.

Slots left for RP	Maximum length of packet sent by master	Maximum length of packet sent by slave
2	1	1
4	1	1
6	3	3
8	3	3

slave's packet length does not have to be restricted). Thus, if the slots left for the RP is 2, the master can send a packet of maximum length = 1 and the gateway can send a packet of maximum length = 1 and so on. Note that for reasons of fairness, the maximum packet length for the master and the gateway is the same. Since the master needs to restrict the maximum length of the gateway's packet to either 1 or 3 (as shown in table 3), we need 2 packet types to achieve this. This procedure effectively suspends the polling of a slave to honor a RP with a gateway. The polling of the slave continues after the gateway has been polled.

In addition, a gateway may lose a slot in switching from one piconet to another. This loss is unavoidable since piconets are in general, not synchronized in time. In the experiments in the paper, we set the value of the threshold to three times the payload of a DH5 packet, which can give a switching loss of about 3% at heavy loads (every $2 \cdot$threshold slots, the gateway loses about one slot in switching). At light loads, this switching loss does not lead to inefficiency since the sum of the fair shares of the gateway in all its piconets is less than 1 and even after the switching loss, the gateway is able to obtain its fair share. The simulations in the next section do not take this switching loss into account and thus, the bandwidth received by the gateway under heavy loads will be a little smaller than the one shown in the results.

4.3. Proof of fairness

We now prove that the above algorithm leads to a max-min fair distribution of the bandwidth of a scatternet among units. We start by proving this in the case of a piconet. In the next step, we will extend the proof to the general case of a scatternet.

4.3.1. Fairness in a piconet

Let us introduce the following notation:

- S: number of slave units in the piconet;
- g_i: rate-demand of the ith unit;
- $\overline{\eta_i}$: rate achieved by the ith unit;
- $\overline{r_i}$: rate-estimation of the ith unit (as defined in equation (2)),

where $\overline{\eta_i}$ and $\overline{r_i}$ are average values.

Slave unit i is referred to as "satisfied", if it achieves it rate demand, i.e., $\overline{\eta_i} = g_i$; else, the slave unit is referred to as "unsatisfied". Also, in the proof that follows, "slot" refers to "Bluetooth slot"; "unit" and "slave unit" may be used interchangeably.

If there is one slave unit in a piconet, then it will always get polled and hence, the algorithm is fair. We prove the fairness when there are two or more slave units.

We first make the following observations:

(a) If a unit has a rate-estimation, $r \geqslant 0.25$, it will never achieve a lesser rate than any other unit.

r is an estimation of the average number of slots of traffic that a master–slave pair will generate per slot in each direction. Thus, a rate of 0.25 means that a master–slave pair generates, on the average, "threshold" slots of traffic in each direction in every $4 \cdot$ threshold slots. Suppose a piconet has two slaves, and the first has a rate-estimation, $r \geqslant 0.25$, then the first slave will be polled at least once in every $4 \cdot$threshold slots, i.e., will get on the average at least threshold polling slots out of every $2 \cdot$ threshold, regardless of the r of the other slave (since N increases at the rate of r, N will increase by at least $0.25 \cdot 4 \cdot$ threshold = threshold; thus, the slave will enter into the "active list" in $4 \cdot$ threshold slots). Thus, it will never achieve a lesser rate than another unit. It is easy to see that this property would be true if there were more than two slaves (two slaves is the worst case).

(b) For $\delta \geqslant 0.1$ and $\alpha \geqslant 0.6$, an unsatisfied slave will tend to a rate-estimation of at least 0.25.

For an unsatisfied slave, the second part of equation (2) (when $x = M$) is always used for updating the rate. Thus, if r_i is the ith rate-estimation:

$$r_{n+1} = \alpha r_n + (1 - \alpha)\frac{M}{T_n} + \delta.$$

This leads to (as n becomes very large):

$$r = (1 - \alpha)M \sum_{k=0}^{\infty} \frac{\alpha^{n-k}}{T_k} + \frac{\delta}{1 - \alpha} \geqslant \frac{\delta}{1 - \alpha}.$$

Thus, for $\delta \geqslant 0.1$ and $\alpha \geqslant 0.6$, for any value of T, the value of r tends to at least 0.25.

(c) As long as there is an unsatisfied unit, the utilization of the system capacity is 1 (for $\delta \geqslant 0.15$ and $\alpha \geqslant 0.65$).

Consider a piconet consisting of seven slave units, in which the first unit, unit_1 is unsatisfied. From (a) and (b), unit_1 will never achieve a lesser rate than any other unit; this means that it will be polled at least once for each time the other slaves are polled. The value of T (as in equation (2)) for unit_1 is thus, at most, $14 \cdot$ threshold. For this value of T and for $\delta = 0.15$ and $\alpha = 0.65$, r for unit_1 tends to at least 0.5. A value of $r = 0.5$ for a slave unit means that it can be polled all the time (since N increases at the rate of r, N will increase by at least $0.5 \cdot 2 \cdot$ threshold = threshold; thus, the slave will enter into the "active list" in $2 \cdot$ threshold slots, which is also the time of its polling). Thus, the system capacity is totally utilized. If there were less than 7 slave units, the value of T would be smaller (than $14 \cdot$ threshold), and r would tend to a higher value (than 0.5).

We choose values of δ and α to satisfy the above properties, i.e., $\delta = 0.15$ and $\alpha = 0.65$.

The following statements hold.

(i) Units with the same rate-demand achieve the same average rate:

$$\overline{g_i} = \overline{g_j} \quad \Rightarrow \quad \overline{\eta_i} = \overline{\eta_j}.$$

We prove this by contradiction. Suppose there are two units, unit_1 and unit_2 with rate demands g_1 and g_2, respectively, such that $\overline{g_1} = \overline{g_2}$. Also, suppose one unit achieves a higher average rate than the other, $\overline{\eta_1} > \overline{\eta_2}$.

Now, unit_2 does not achieve its rate-demand (since $\overline{\eta_1} > \overline{\eta_2}$). Unit_1 may or may not achieve its rate demand. From property (b), unit_2 will always tend to a value at least equal to 0.25, since it is an unsatisfied slave. Using property (a), this implies that $\overline{\eta_2}$ cannot be less than $\overline{\eta_1}$. This is a contradiction.

(ii) Units with a higher rate-demand achieve an average rate at least equal to that achieved by units with a lower rate-demand:

$$\overline{g_i} > \overline{g_j} \quad \Rightarrow \quad \overline{\eta_i} \geqslant \overline{\eta_j}.$$

This can be proved by contradiction in the same manner as in part (i).

Now, without loss of generality, let us partition the slave units into two sets, $S1$ and $S2$, in such a way that units in $S1$ are satisfied, while units in $S2$ are not.

- If the set $S2$ is empty, than all the units achieve their rate-demand and the system is fair.
- If the set $S2$ is not empty, then using statements (i) and (ii), all units share the bandwidth in a fair manner. Moreover, since $S2$ contains at least one unit, the total system capacity is utilized. Hence, it is not possible to increase the rate of a unit in $S2$ without decreasing the rate of some other unit.

4.3.2. Fairness in a scatternet

The proof of fairness for a scatternet follows trivially from that for a piconet. We make the following two observations:

(1) The gateway visits a piconet only after the estimation of N for the piconet becomes greater than the threshold (it calculates N_{thresh} while determining the next RP). In other words, the "virtual master" (gateway) does not poll (visit) its "virtual slave" (master) till the estimate of N becomes greater than the threshold. This is similar to the algorithm used by the master to poll the slaves in which a slave is not polled till its estimate of N becomes greater than the threshold. Thus, the gateway divides its presence among its piconets in a fair manner, i.e., according to the SPF. Note that if the PPF for a gateway in a piconet is less than its SPF, the master does not poll the gateway for more than the PPF. Thus, the apparent rate demand and SPF for the gateway in the piconet are reduced. This may increase the SPF of the gateway in other piconets. In this case, the gateway divides its presence according to the updated SPFs.

(2) While calculating the next RP for a gateway, the master calculates the *num_slots* value which estimates the number of slaves in one polling cycle (starting from the slave after the gateway in the polling cycle) who will have their values of N greater than the threshold at the estimated time of their poll. This achieves fairness between the gateway and the non-gateway slaves. Also, the master continues to use the same algorithm for polling non-gateway slaves in a scatternet as described for a piconet in section 4.1. This maintains fairness between non-gateway slaves, i.e., the division is done according to the PPFs (or the updated PPFs).

4.4. Overhead/limitations of the algorithm

The rate calculations will lead to a higher load on the system. Also, the algorithm does not take into account SCO links. We believe (and as has been shown in [6]) that ACL links are capable of carrying voice with small delays. The controlled channel access in Bluetooth can ensure good support of voice using ACL links. Also, scheduling in a scatternet where SCO links are allowed may not be feasible. Since SCO links require a periodic reservation of two slots every two, four or six slots, meeting the demands of such a link with a gateway may be impossible when the gateway is visiting some other piconet.

5. Experiments and results

In this section, we present simulation results, which show that the algorithm satisfies the fairness criteria described earlier. We start with simple topologies that illustrate the behavior of the algorithm and then show that it also works well in more complex topologies. There are three topologies that the experiments focus on and these demonstrate the behavior of the algorithm – a topology with (a) a gateway belonging to two piconets, (b) a gateway belonging to three piconets and (c) a piconet having two gateways. The experiments also show the adaptivity of the algorithm, i.e., how quickly the algorithm adapts to changing traffic demands of slaves.

In the experiments, we specify the "rate of a slave", which refers to the sum of the rates at which a slave generates data for a master (i.e., the rate demand of a slave) and the master generates data for the slave. Moreover, unless mentioned otherwise, we assume that the traffic rate from a slave to a master is equal to that from the master to the slave. Thus, a slave having a rate of 0.4 means that the slave generates data at the rate of 0.2 Bluetooth slots per slot and the master also has a rate demand of 0.2 towards the slave. As we show in the section on asymmetric traffic, the algorithm works well even if these two rates are not the same.

The simulation environment used in our experiments is NS-2 [8]. We have augmented NS-2 with the Bluetooth model. The simulator models the Bluetooth baseband, LMP and L2CAP layers and enables the creation of piconets and scatternets. The model contains most of the standard features of Bluetooth like Frequency Hopping, Multi-Slot Packets, Fast ARQ (Automatic Retransmission Query). Note that as mentioned earlier, in our simulator, the switching loss associated with the gateway moving from one piconet to another

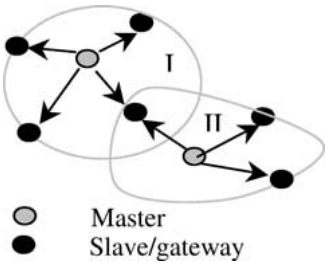

Figure 5. Example scatternet.

is not taken into account. This effect can lead to the gateway losing up to 3% of slots at heavy loads. The experiment results are thus, a slight overestimate.

In the experiments, all traffic generated is CBR. Each experiment is run for a system time of 32 sec. In the experiments, the term "slave" refers to a non-gateway slave; a gateway slave is referred to as "gateway". Also, in experiments where the PPF and the SPF values (and not the updated PPF and the updated SPF) are shown, the PPF and the updated PPF are equal and the SPF and the updated SPF are also equal. In the graphs, "BW" in the index stands for bandwidth, "GW" stands for gateway.

5.1. Single gateway in two piconets

We first consider the simple topology shown in figure 5, which consists of two piconets, numbered I and II, connected by a single gateway. We consider various cases by changing the traffic and the number of slaves in the piconets.

Experiment 1. Adaptation between gateway and non-gateway slave traffic
Each piconet has one non-gateway slave that generates very high traffic, with rate equal to 1, to the master. The gateway has equal traffic to both masters. We vary the gateway traffic to show the fair sharing of the piconet bandwidth between the gateway and the slave. We show the results for one piconet since the two piconets are exactly symmetric.

Figure 6(a) shows the sharing of bandwidth between the gateway and slave for different values of gateway traffic. It also shows the fair share of the slave and the total fraction of the bandwidth obtained by the gateway and the slave in the piconet. It can be seen that the slave obtains a bandwidth equal to its fair share for different values of gateway traffic. Moreover, the sum of the bandwidths obtained by the slave and the gateway is nearly equal to 1. The reason for this to be slightly less than 1 is that some of the piconet capacity is used in sending LMP_hold_req PDUs of the LMP layer.

In figure 6(b), the comparison of the fraction of the bandwidth obtained by the gateway to its SPF (PPF and SPF are equal) is shown. Figure 6(b) shows that the gateway gets almost equal to its fair share of the bandwidth for all values of traffic. Again, the reason that the gateway obtains slightly less than its fair share is because some of the slots are used for LMP PDUs. This also explains why the gateway obtains slightly less than the slave in figure 6(a).

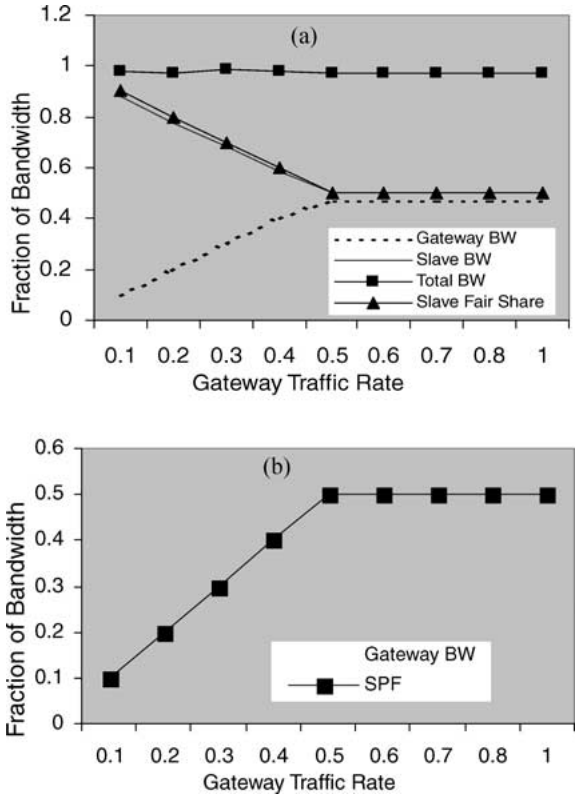

Figure 6. (a) Sharing of bandwidth between gateway and slave. (b) Comparison of fraction of bandwidth obtained to SPF for the gateway.

Experiment 2. Different traffic to piconets
The same topology as in the previous case, but each slave has a traffic rate of 0.3 to the master. The gateway has a fixed traffic rate of 0.2 to the master of Piconet I and variable traffic to the other master. The PPF and SPF of the gateway in the first piconet are, thus, both equal to 0.2. The traffic in Piconet I does not change and the gateway and the slave get a constant fraction of 0.2 and 0.3 of the piconet bandwidth, respectively.

Figure 7(a) shows the sharing of bandwidth between the gateway and slave for different values of gateway traffic, while figure 7(b) shows the comparison of the fraction of the bandwidth obtained by the gateway in Piconet II to its SPF and PPF. From the graphs, we can see that when the gateway has different traffic to piconets, it divides its presence among the piconets according to the traffic offered and in a fair manner (again, the gateway obtains slightly less than its fair share due to the LMP PDUs). Also, the gateway makes use of the lower traffic offered by the slave in Piconet II to obtain a higher share of the bandwidth in Piconet II.

Experiment 3. Different number of slaves
Piconet I has 3 slaves, while the number of slaves in Piconet II is variable. Each slave generates traffic to the master at the rate of 0.2. The gateway has a traffic rate of 0.3 to Piconet I and 0.8 to Piconet II. The PPF and SPF of the gateway in Piconet I are, thus, 0.2 and 0.3, respectively. In Piconet II, the value of PPF changes depending upon the number of slaves.

In Piconet I, the slaves get a bandwidth fraction of 0.2 and the gateway gets 0.3. Figure 8(a) shows the sharing

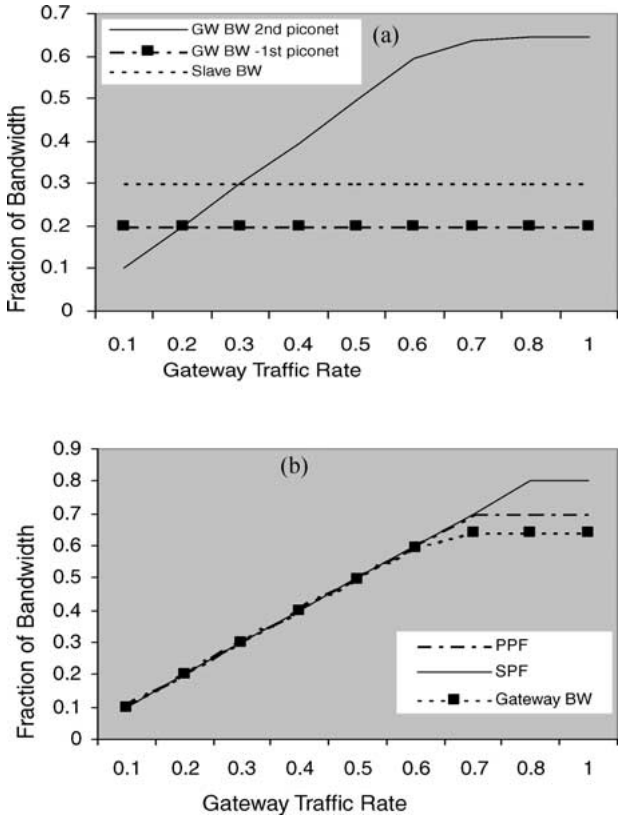

Figure 7. (a) Sharing of bandwidth between gateway and slave in Piconet II. (b) Comparison of fraction of bandwidth obtained by the gateway to SPF and PPF in Piconet II.

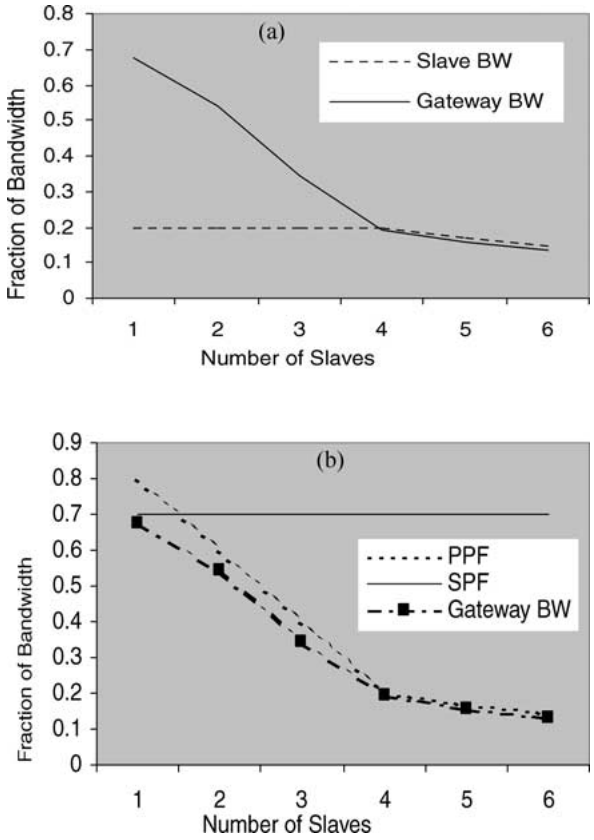

Figure 8. (a) Sharing of bandwidth between gateway and slave in Piconet II. (b) Comparison of fraction of bandwidth obtained by the gateway to SPF and PPF in Piconet II.

of bandwidth between the gateway and each slave in Piconet II. Figure 8(b) shows the comparison of the fraction of the bandwidth obtained by the gateway in Piconet II to the SPF and PPF. The gateway receives a fraction of the bandwidth almost equal to its fair share. Also, as the number of slaves increases, the fraction of the bandwidth received by the gateway (and each slave) reduces in a fair manner.

Experiment 4. Asymmetric traffic

We now consider a case where the traffic rates from Master to Slave and Slave to Master are different (asymmetric traffic). We consider the same topology as in experiment 2 of the current section, with the non-gateway slaves having the same rate as in experiment 2. The gateway has a fixed traffic rate of 0.2 to the master of Piconet I and variable traffic to the other master. The variable traffic is such that traffic from Master to Slave has a rate of 0.1 and traffic from Slave to Master varies.

Figure 9 shows the comparison of bandwidth fraction obtained by the gateway in this experiment versus that obtained by the gateway in experiment 2 in Piconet II for different values of gateway traffic (which is the sum of master to gateway and gateway to master traffic rates). We see that the fraction is slightly lower than the fraction obtained in experiment 2. Asymmetric traffic leads to wastage of slots, since an empty slot is returned in one direction where there is no data to send. It can be seen though, that the gateway still behaves in an approximately fair manner. All other bandwidth fractions for slaves and the gateway are the same as in experiment 2.

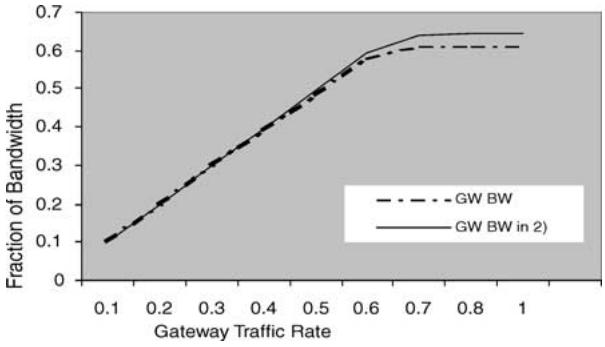

Figure 9. Comparison of fraction of bandwidth obtained by gateway in this experiment with that in experiment 2 in Piconet II.

5.2. Single gateway shared between three piconets

We now consider a topology, where a gateway is shared between 3 piconets, numbered I, II and III. Piconet I has 5, Piconet II has 1 and Piconet III has 4 slaves. Each slave has a traffic rate of 0.2. The gateway has a traffic rate of 0.2 to Piconet I, 0.3 to Piconet III and a variable rate to Piconet II. All traffic is symmetric (same from master to slave and from slave to master).

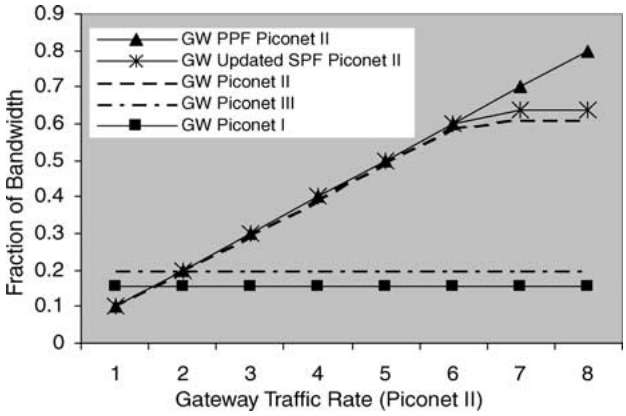

Figure 10. Bandwidth fraction received by gateway in the three piconets.

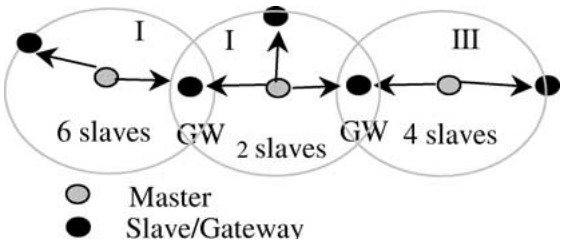

Figure 11. Example scatternet topology.

Figure 10 shows the fraction of bandwidth obtained by the gateway in each piconet with increasing gateway traffic rate to Piconet II. It also shows the PPF and the Updated SPF of the gateway in Piconet II. We do not show the fair shares of the gateway in Piconet I and III since they are constant (0.16 and 0.2, respectively). It can be seen that the gateway manages to get close to its fair share in the 3 piconets. The slaves in Piconet I get a bandwidth fraction of 0.16 and the slaves in Piconet II and III get a bandwidth fraction of 0.2 (all these are equal to their fair shares).

5.3. Piconet with two gateways

We now show the working of the algorithm in a piconet having 2 gateways, as shown in figure 11. Piconets I, II and III have 6, 2 and 4 non-gateway slaves, respectively. There are two gateways, GW 1 between Piconets I and II; and GW 2 between Piconets II and III. All slaves have a traffic rate of 0.2. GW 1 has a traffic rate of 0.2 in Piconet I and 0.5 in Piconet II. GW 2 has a traffic rate of 0.2 in Piconet III. We vary the traffic rate of GW 2 in Piconet II and show the fair sharing of bandwidth.

Figure 12 shows the fraction of bandwidth obtained by GW 1 and GW 2 in Piconet II compared to their fair shares. The x-axis denotes GW 2 traffic in Piconet II. It can be seen that the bandwidth fractions obtained are very close to the fair value. The non-gateway slaves of Piconet II receive a bandwidth fraction of 0.2, which is equal to their fair share (not shown in the figure). The bandwidth fraction received by slaves in Piconets I and III does not change for different values of GW2 traffic in Piconet II. The fair share of each slave (including the gateway) in Piconet I is 0.14 and in Piconet III

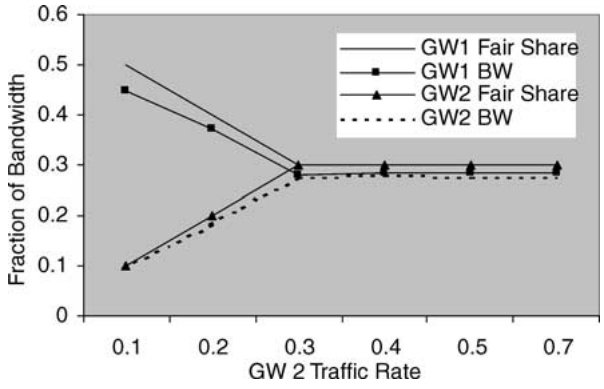

Figure 12. Fraction of bandwidth and fair share of GW1 and GW2 in Piconet II.

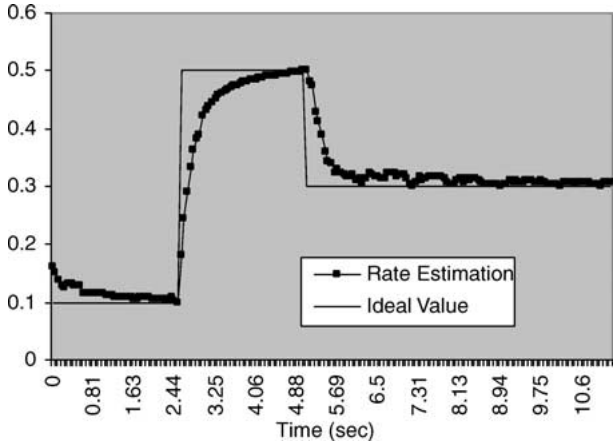

Figure 13. Actual rate estimation of the gateway and its ideal value.

is 0.2; the bandwidth fraction received by each slave is very close to these fair shares.

5.4. Adaptivity to changing traffic demands

We now show how quickly the algorithm is able to adapt to changing traffic. We again consider the scenario of experiment 1 of section 5.1, consisting of two piconets, each having a non-gateway slave, connected by a single gateway. The non-gateway slaves have a traffic rate of 1; the gateway has equal traffic to both the masters. We vary the traffic rate of the gateway as time progresses: for the first 2.5 seconds, the gateway's rate is 0.1, for the next 2.5 seconds, it is 0.5 and for the remaining time, it is 0.3.

Figure 13 shows the actual rate estimation of the gateway (and its ideal value) versus time. It can be seen that the rate estimation adapts very quickly to the new rate. For example, when the rate changes from 0.1 to 0.5, the rate estimation reaches a value of 0.45 in about half a second after 2.5 sec. Thus, the algorithm adapts to quickly changing traffic.

6. Conclusions

This paper proposed a distributed scatternet-scheduling algorithm that adapts to non-uniform and changing traffic. This

algorithm provides an integrated solution for both intra- and inter-piconet scheduling and can be implemented using the HOLD mode of Bluetooth. Through analysis and simulations, we showed that the algorithm is traffic-adaptive and results in a fair allocation of bandwidth to units. We explained earlier that the algorithm may allow a unit to go into a power-saving mode.

In future, we would like to explore this option, which also assumes importance since Bluetooth devices will most likely operate in a power-constrained environment. As future work, we would also like to evaluate the performance of TCP and other kinds of traffic on our algorithm. We are also working towards interfacing the algorithm with requirements of higher layers. In this respect, we are working towards providing QoS support using the algorithm.

References

[1] S. Baatz, M. Frank, C. Kehl, P. Martini and C. Scholz, Adaptive scatternet support for Bluetooth using sniff mode, in: *Proc. of IEEE LCN* (2001).

[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, in: *Proc. of IEEE INFOCOM'2001* (2001).

[3] J. Haartsen, BLUETOOTH – the universal radio interface for ad hoc wireless connectivity, Ericsson Review 3 (1998) 110–117.

[4] P. Johansson, M. Kazantzidis, R. Kapoor and M. Gerla, Bluetooth – an enabler for personal area networking, IEEE Network Magazine, Wireless Personal Area Network (September 2001).

[5] M. Kalia, D. Bansal and R. Shorey, MAC scheduling and SAR policies for Bluetooth: A master driven TDD pico-cellular wireless system, in: *Proc. of 6th IEEE International Workshop on Mobile Multimedia Communications (MOMUC)* (1999).

[6] R. Kapoor, L. Chen, Y. Lee and M. Gerla, Bluetooth: carrying voice over ACL links, in: *Proc. of MWCN* (2002).

[7] A. Mayer, Y. Ofek and M. Yung, Approximating max-min fair rates via distributed local scheduling with partial information, in: *Proc. of IEEE INFOCOM* (1996).

[8] NS-2 simulator, `http://www.isi.edu/nsnam/ns/`

[9] A. Racz, G. Miklos, F. Kubinszky and A. Valko, A pseudo-random coordinated scheduling algorithm for Bluetooth scatternets, in: *Proc. of MobiHoc* (2001).

[10] Specifications of the Bluetooth System – core, Vol. 1, v. 1.1, `www.Bluetooth.com`

[11] W. Zhang and G. Cao, A flexible scatternet-wide scheduling algorithm for Bluetooth networks, in: *Proc. of IEEE IPCCC* (2002).

Rohit Kapoor received his Bachelor degree in computer science in 1999 from the University of Roorkee, India. He is currently a Ph.D. candidate at the University of California, Los Angeles (UCLA). His research focuses on Bluetooth-based personal area networks. He is a member of the Network Research Lab at UCLA.
E-mail: rohitk@cs.ucla.edu

Andrea Zanella received the Ph.D. degree in telecommunication engineering from the University of Padova, Italy, in 2002. Prior to that he received the Dr. Ing. degree (comparable to Master degree) in computer engineering in 1998, still from the University of Padova. He spent nine months, in 2001, as post-doc researcher at the Department of Computer Science of the University of California, Los Angeles (UCLA), where he was engaged in research on Wireless Networks and Wireless Access to Internet under the supervision of Prof. Mario Gerla. Currently, he is a research fellow in the Department of Information Engineering of the University of Padova, Italy. His research interests are mainly focused on topics related to wireless and mobile networking. In particular, in the last period, he has been working on the performance aspects of wireless personal area networks based on the Bluetooth standard.
E-mail: zanella@dei.unipd.it

Mario Gerla is a professor in the Computer Science Department at UCLA. He received his graduate degree in engineering from the Politecnico di Milano in 1966, and his M.S. and Ph.D. degrees in engineering from UCLA in 1970 and 1973, respectively. He joined the faculty of the UCLA Computer Science Department in 1977. His current research is in the area of analysis, design and control of communication networks. Ongoing projects include the design and evaluation of QoS routing and multicast algorithms for IP domains, the design and evaluation of all-optical network topologies and access protocols, the design of wireless mobile, multimedia networks for mobile computing applications, and the development of measurement methods and tools for evaluating the performance of high-speed networks and applications.
E-mail: gerla@cs.ucla.edu