Adaptive Scatternet Support for Bluetooth using Sniff Mode

Simon Baatz¹, Matthias Frank¹, Carmen Kühl², Peter Martini¹, Christoph Scholz¹

¹ University of Bonn, Institute of Computer Science IV, Roemerstrasse 164, D-53117 Bonn, Germany
{baatz, matthew, martini, scholz}@cs.uni-bonn.de

² Nokia Mobile Phones Produktionsgesellschaft mbH, R&D Center Germany, P.O. Box 101823, D-44718 Bochum, Germany
carmen.kuhl@nokia.com

Abstract
Future applications of Bluetooth are likely to include ad-hoc networking. Therefore, it is desirable to interconnect multiple Bluetooth picocnets to form a scatternet. Up to now, there is no extensive proposal for scatternet support available. In this paper, we present and analyze an adaptive scheme for scatternet scheduling that bases on sniff mode and thus does not require substantial modification of the current Bluetooth specification. The suitability of our approach is shown by first simulation results.

Keywords: Bluetooth, Ad Hoc Networks, Wireless Networks, Scatternet, Sniff Mode, Scheduling

1. Introduction
While the market for fixed, wired personal computers shows stagnating tendencies, mobile computing receives more and more attention. The terms ubiquitous computing and pervasive computing have moved into the center of interest. Truly mobile devices such as mobile phones, digital watches, and personal digital assistants (PDA) become more and more powerful. At the same time, notebook devices are decreasing in size and weight. As the Internet becomes a part of everyday life for many people, there is not only the need for pervasive computing but also for pervasive networking.

Up to now, users that want to connect for example their notebook to a mobile phone have to employ expensive, proprietary cabling or use infrared communication technology that requires devices to have a free line of sight between them. Bluetooth ([11], [2]) is a radio technology that promises to be a very convenient, low-cost solution for the interconnection of all kinds of mobile devices.

This cheap but powerful technology will be available in a wide range of mobile devices. Thus, building Bluetooth ad hoc networks connecting all sorts of devices will be a common application for Bluetooth devices on conferences, meetings, and so on. Up to now, Bluetooth defines a simple network topology (piconet) that only supports a limited number of devices and requires all devices to be in range. However, support of the more demanding multi-hop ad hoc network topologies (so-called scatternets) is not specified in detail yet. This paper proposes a mechanism to support scatternet communication by (re)using mechanisms currently specified and thus provides the basis for Bluetooth ad-hoc networking.

The paper is structured as follows. Section 2 gives a short summary of the Bluetooth technology. In section 3, basic properties of Bluetooth scatternet support are presented thereby explaining its demands and challenges. Section 4 features our adaptive approach to scatternet scheduling. Section 5 illustrates the properties of our approach by presenting simulation results in basic scenarios. Finally, section 6 presents our conclusion and describes future work.

2. Bluetooth Overview
Bluetooth is a low-cost, low-power wireless networking technology designed to be used in a person’s operating space, i.e. the space around a person that typically extends up to 10 meters in all directions. The Bluetooth radio transmission uses a slotted protocol with a FHSS (Frequency Hopping Spread Spectrum) technique in the globally available and unlicensed 2.4 GHz ISM (Industrial, Scientific and Medical) band. The hop frequency is up to 1600 hops per second, the frequency spectrum is divided into 79 channels (or 23 in some countries) of 1 MHz bandwidth each. The frequency hopping scheme is combined with fast ARQ (Automatic Repeat Request), CRC (Cyclic Redundancy Check) and FEC (Forward Error Correction) to achieve appropriate reliability on the wireless link. Communication of Bluetooth devices follows a strict master-slave scheme, i.e. there is no way for slave devices to communicate directly with each other. A device acting as a master can connect to a maximum of 7 active slaves. Master and slaves form a so-called piconet.
(cf. Figure 1), in which the master defines the timing and
the hop pattern. The slaves have to stay synchronized to
the master while participating in the piconet.

![Diagram of a piconet with a master and slaves]

**Figure 1: A piconet and a scatternet consisting
of three piconets**

Multiple piconets may be connected to form a so-called scatternet, a multi-hop Bluetooth network. In such scatternets, devices that participate in multiple piconets connect the piconets with each other. These connecting devices may have a master role in a piconet of their own and slave roles in one or more other piconets. Figure 1 shows two connecting devices, a master/slave device and a slave/slave device interconnecting two piconets each.

### 2.1. Time Division Duplex

Between master-slave pairs, both Synchronous Connection Oriented (SCO) links (typically used for voice) and an Asynchronous Connectionless link (ACL) are supported, the latter ones being dealt with in this paper. While SCO communication takes place on a regular basis in previously reserved slots, a Time Division Duplex (TDD) scheme is applied to control access of ACL links to the remaining (unreserved) bandwidth. The master may begin to send a packet in even numbered slots only. The slave addressed by this packet is the only device allowed to send in the odd numbered slot following the master’s packet. In order to keep up the alternation between even and odd slots, packets must occupy an odd number of slots. Consequently, Bluetooth defines different packet types with a length of 1, 3, or 5 slots. In this paper, we only consider packets with no FEC for the payload, which are called DH1, DH3, and DH5 depending on the amount of slots they occupy. If a master wants to permit a slave to send without having data for the slave, it may send special 1-slot packets without payload (POLL or NULL packets). In contrast to the NULL packet, a POLL packet forces the slave to respond with a packet. A slave that needs to respond to a master transmission and has no data ready for transmission will use a NULL packet. In order to guarantee a minimal quality of service, Bluetooth allows each master/slave pair to agree on a maximum interval $T_{\text{poll}}$ (measured in slots) between two consecutive polls by the master.

![Diagram of TDD medium access control]

**Figure 2: Time Division Duplex access scheme**

An example for the TDD medium access control is depicted in Figure 2. The piconet master sends a three slot packet to the second of its two slaves in the slots 0 to 2. The addressed slave may respond in the subsequent slot 3. In most cases, it will respond at least with a packet that acknowledges the master’s transmission. As – in our example – the packet sent by the second slave occupies only one slot, the master is free to address slave 1 in slot 4. If the master does not perform any transmission like in slot 6, no slave is allowed to send in the subsequent odd numbered slot.

### 2.2. Low Power Modes

Being a technology optimized for portable devices with constrained power resources, Bluetooth offers various power saving modes which are used to reduce the duty cycle of devices: hold, park and sniff mode. In the following, only sniff mode is introduced, as it is used to realize the proposed scatternet communication scheme.

The basic idea of sniff mode is to reduce the duty cycle on a link between two devices by negotiating specific slots (sniff slots) where communication between devices can begin. If no communication takes place at these slots, the devices may spend the time until the next sniff slots in a low power mode. Otherwise, the communication period (sniff event) may be extended dynamically, until one of the devices decides to end the communication. The other device aborts the communication if it does not receive anything on the link for a configurable amount of slots$^1$.

---

$^1$ In fact, this behavior is specified only for the slave. However, if a master does not receive anything from a slave for some time (e.g. due to transmission errors), it has to assume that the slave has already gone back to low power state.
Figure 3 shows the activity on a link in sniff mode. Sniff slots are equally spaced with a period called $T_{\text{sniff}}$. In this example, a single sniff slot is used per sniff period. At the first sniff slot shown, the master does not use the sniff slot (by sending a packet). However, the slave uses this sniff slot by listening to the master. As the slave is not addressed in this slot, it goes back to low power state immediately.

**Figure 3: Activity on a link in sniff mode**

The master addresses the slave in the next sniff slot; however the transmission fails due to a transmission error and the slave does not respond. As the master knows that the slave listens only for a single sniff slot, it does not try to reach the slave in the following even slot. Instead, it has to wait until the next sniff slot occurs, where the two devices meet, i.e. the master sends a packet while the slave listens at the sniff slot and the slave responds in the following odd slot. Once they have met, they communicate with each other until one of the devices (the master in this example) decides to terminate communication. The slave reenters power save state after the predefined time-out occurs without having received anything from the master, as can be seen in Figure 3.

3. Challenges in Scatternet Support for Bluetooth

The current Bluetooth specification already defines scatternets. However, it focuses on single piconet operation and leaves open many details of scatternet operation. Performance of and scheduling in scatternets has already been studied in a generic form in [3] and [4], not addressing effects like clock drift and the implications of complex scatternet topologies sufficiently. In this section, we present some of the difficulties underlying scatternet support, which are important to understand the design decisions made for our approach.

3.1. Switching Piconets

If a device is connected to multiple piconets, these piconets will use different frequencies most of the time, since their hop selection and timing are independent of each other. In fact, this is the reason why a scatternet offers higher aggregate bandwidth than a single piconet (cf. [5] for a first study of the performance aspects of Bluetooth scatternet topologies).

Devices will typically be equipped with a single Bluetooth transceiver, which will be able to follow the hopping scheme of a single piconet at a time. Apart from the interference problems to be addressed when integrating several transceivers in a compact design, this takes costs and power consumption too high for the majority of mobile devices. Consequently, a device has to switch between the piconets it is part of in a time division multiplex manner.

**Figure 4: Piconet switching losses**

With the master clocks defining the piconet timings and these clocks not being synchronized to each other, the slot boundaries of different piconets do not match in general (cf. Figure 4). After a piconet switch, a device has to wait at least until the next even slot begins in order to be able to participate in the piconet. Thus, each piconet switch may cost up to two slots (so-called guard time), which cannot be used for communication.

As a consequence, piconet switches should not happen too often. Conversely, a low piconet switching frequency causes high forwarding delays with respect to data connections crossing several hops. Therefore, a compromise must be found between bandwidth usage and maximum delay.

3.2. Scheduling Switches

In order to realize time division multiplexing between piconets, the current Bluetooth specification suggests to put links in power save mode in order to “leave” a piconet and to be able to visit another piconet.

**Figure 5: Scatternet with 3 scheduling phases**

As described above, two devices of a specific link can only communicate if both devices take part in the same
piconet at the same time. This requires some kind of coordination between devices. Ideally, each device knows exactly when its peer device is available, i.e. is present in the proper piconet. Negotiation and distribution of piconet presence schedules is not as easy as it seems at first glance. As an example, consider the scatternet shown in the left part of Figure 5 where each device has two links to other devices and participates in two piconets. In the simplest case, each device would try to divide its airtime evenly between its two links and coordinate with its peer devices on the timing of the piconet presence.

When taking a closer look at the topology, it becomes clear that such a schedule cannot exist. Links 1 and 2 cannot be active at the same time as device A has to switch between these links. This is also true for links 2 and 3 (because of device B) and 3 and 1 (because of device C). This mutual exclusion may be modeled by the graph shown in the right half of Figure 5, where the minimum coloring reflects the amount of different piconet presence assignments that is needed in order to be able to serve all links. In the graph, each node represents a link and each edge represents a mutual exclusion of two links induced by a certain device (i.e. if a device forces mutual exclusion between \( n \) links, it is represented as an \( n \)-clique). Obviously, the minimum coloring requires 3 colors meaning that only one link can be active at a time. If the links are treated equally, each device has to divide its airtime in thirds, as shown by Figure 6, and only 2/3 of each device’s airtime can be used.

3.3. Adaptivity

Another point making static piconet presence schedules throughout scatternets inappropriate is their inflexibility under varying load. In the schedule discussed above only one third of the airtime can be used for a link, even if the other two links are idle. As before, changing the schedule of a single link may have implications on the whole scatternet.

In order to achieve adaptivity, frequent recalculations of the schedule would be required based on the utilization of the links.

3.4. Clock Drift

Each master has a clock determining the timing of its piconet. As this clock is free-running, the clock speed may differ slightly between different masters. Bluetooth allows a clock drift of max. ± 20 ppm (parts per million) during activity, and 250 ppm during low power modes. Therefore, two piconets may drift against each other with up to 40 ppm in active mode, i.e. 2 slots approx. every 30 seconds.

This requires constant monitoring of the clock drift and adaptation of the schedules based on the drift.

4. Credit Based Scatternet Scheduling

As pointed out in the previous section, calculating and maintaining piconet presence schedules for the whole scatternet is likely to require too many resources. Additionally, this approach reacts rather slowly to the considerable load variations that are common in typical data traffic.

As an alternative, agreeing on entirely local communication schedules may waste much bandwidth, as the local piconet presence phases may overlap or may include unnecessary idle time.

For these reasons, our approach does not use strictly defined local communication schedules. Instead, it defines presence points for each inter-piconet link at which communication may start. The crucial point about these presence points is to enable each device to quickly determine whether the peer device is in the same piconet. If so, the communication may start between the devices. Otherwise, another presence point may be tried without having lost much bandwidth. The length of a particular communication period is not predetermined, as it depends on the current link utilization and the amount of data ready to exchange. Interestingly, the presence points and the dynamic length of communication periods may be mapped directly onto sniff mode, requiring no changes or almost no changes to the current specification.

The communication schedule is determined online (i.e. not a priori) for each communication period. Therefore,
the decisions when to use a presence point and how long
to stay in a certain piconet must be based on simple cal-
culations. As no fixed schedules exist, a device can cope
with clock drift simply by monitoring the timing of its
masters’ presence points. Of course, interference will
occur when operating scatternets. Consequently, missing
a presence point must have low impact, and thus presence
points should be placed rather dense.

In the following sections, we will present the details of
our scatternet scheduling approach using the presence
point concept.

4.1. Mapping the presence point concept onto
sniff mode

Since communication between two devices continues
as long as data is exchanged in sniff mode, it has a dy-
namic nature that makes it suitable for scatternet sched-
uling. The sniff slots are regarded as possible presence
points at which peer devices may start communicating.
However, both devices may skip sniff slots at their own
discretion. Thus, the probability of two devices using a
common sniff slot has to be maximized.

In addition to the sniff parameters, the maximum poll
interval $T_{\text{poll}}$ is employed to place an upper bound to the
time between two subsequent sniff events, thereby pro-
viding means for decent quality of service.

Two devices on a link stay with an ongoing sniff event
until there is no more data to be sent or until one or both
of the devices want to abort the current sniff event in
order to use a sniff slot of another link. As noted above, a
device recognizes an abortion by the peer after a timeout.

If neither of the devices has any more data to sent, the
slave will react with a NULL packet to the master's POLL
packet. In consequence, a sniff event will be terminated
by both devices when detecting such a POLL-NULL
sequence.

4.2. Credit Scheme

Sniff mode provides the basic mechanism to imple-
ment the presence point concept. However, the crucial
point of the communication scheme is to decide when to
abort an ongoing sniff event in order to use an upcoming
sniff slot. This decision is based on a priority scheme,
which is explained in the following and was inspired by
leaky bucket traffic shaping (cf. [6]) and the Deficit
Round Robin fair scheduling mechanism (cf. [7]).

Each device internally manages a priority for each of
its links. Note that these priorities are only valid per de-
vice and not per link, i.e. the two devices of a link may
assign different priorities to the same link. A link has a
higher priority relative to another link assigned if it has
been treated unfairly in the past when compared to the
other. This relative fairness is quantified by keeping track
of the amount of slots used for each link of a device.
Therefore, a device assigns each of its links a credit ac-
count. Each credit on this account represents the use of
one slot of the corresponding link. Thus, a link that uses
a slot for transmitting or listening is charged one credit.
The intention is to exactly track the number of slots used
by each link in order to be able to serve the links in a fair
manner, even though presence of devices at sniff slots is
not completely predictable. If a slot is not used at all, a
special temporary account is debited. Thus, in each slot
one of the accounts is debited one credit.

Since one credit per slot is leaving the system, one
credit per slot is entering the system in order to keep the
sum of all credit accounts constant. This is achieved by
increasing the temporary account mentioned above by one
credit for each slot. As the algorithm starts with all ac-
counts holding zero credits, the overall sum of all credits
always equals zero.

The credits given to the temporary account are later
distributed to the link's credit accounts. As soon as the
temporary account holds at least $n$ credits (with $n$ being
the number of links the device has), it is decreased by $n$
credits and each link’s credit account is incremented by
one. This distribution mechanism treats the links in an
absolutely fair way since each link gets the same increase
of credit at the same time. As an alternative, links might
get their credits using a more sophisticated scheme taking
care of the links’ QoS demands.

The decision when to use a sniff slot is based on the
values of the credit accounts. An ongoing sniff event is
aborted in order to use another link's upcoming sniff slot
if the latter has a higher number of credits, indicating a
lower relative allocation of airtime for the corresponding
link. Note that the decision to abort a sniff event is not
agreed on between the two devices participating in the
sniff event. The device not actively aborting will notice
the end of the sniff event through timeout.

![Figure 7: Credit scheme example](image)

Figure 7 shows an example for the credit scheme on a
device with three links. The figure depicts the sniff slots,
the activity, and the credit values for each link. Additionally, the credit values for the temporary account are given. In the beginning, a sniff event on link 1 takes place. Consequently, the credit account of link 1 is decremented every slot whereas the credit accounts of links 2 and 3 remain unchanged. The temporary account is incremented every slot. Every 3 slots, its value reaches 3 and the credits are distributed to the links, i.e. each link’s credit account gets a credit. Right before the sniff slot on link 2, the credit value is −1 for link 2 and 2 for link 1. As the credit value for link 1 is higher, the ongoing sniff event is not aborted and link 2 is not attended. The situation is reversed for the sniff slot of link 3, where the ongoing sniff event is aborted in order to attend the sniff slot on link 3 because its credit value is higher than the value for link 1.

An ongoing sniff event is also interrupted if another link is about to reach its poll interval, i.e. if it has not been polled for nearly \( T_{\text{poll}} \) slots. This decision is completely independent from the credit account values. It ensures an upper bound on the time between two consecutive polls of the same link.

In addition to that, a sniff event may end prematurely due to a POLL-NULL sequence or a sniff timeout. In order to minimize the idle time in case of a premature end, a greedy approach was chosen, i.e. the next available sniff slot is tried in order to start a sniff event regardless of the respective link’s credit value.

As sniff slots are coordinated only locally, those on different links may overlap. Thus, it may be necessary to choose one of several possible sniff slots that may be taken next. In this case, links that need to be polled because they reached their poll interval are treated with highest priority. If none of the links reached its poll interval, the link with the highest credit value is taken since it has been treated unfairly when compared to the others.

### 4.3. Switch Threshold

The basic credit scheme presented so far may lead to frequent switches between links and thus between different piconets. As guard time is lost in each piconet switch, it is desirable to minimize the number of switches. It may seem straightforward to guarantee a minimum amount of slots (i.e. a minimum service time) a link is served after it is switched to. However, this may easily lead to starvation of links since sniff slots of a link may constantly occur during other links’ guaranteed minimum service times.

Figure 8 depicts an example of a starvation situation on a device with three links. The first two links have overlapping sniff slots and always get exactly their minimum service time. Thus, the device switches back and forth between these two links as there is always a sniff slot directly after minimum service. As the minimum service period is not interruptible when defined as above, the third link’s sniff slots are never taken even though its priority (i.e. the credit value) is constantly increasing. As a conclusion, a scheme has to be proposed that takes care of priorities.

**Figure 8: Starvation scenario**

We propose a scheme that allows the abortion of an ongoing sniff event only if the credits of the link, which should be switched to, exceed the current link’s credits by a certain threshold called \( N_{\text{switch,th}} \).

The impact of \( N_{\text{switch,th}} \) has been analyzed under the following assumptions. Firstly, it is assumed that all links are saturated. Secondly, only single-slot packets are considered. Thirdly, it is assumed that the peer device does not abort the mutual sniff event prematurely (i.e. before the considered device aborts the sniff event). Furthermore, it is assumed that a sniff event for any link may start in any even slot. For example, these assumptions are valid for the scenario considered in section 0.

Under these assumptions, different initial credit values for the link’s credit accounts result in differing schedules, but all schedules are stable and have common properties. The \( n \) different links of a device experience a round-robin service. Interestingly, there are only \( n−1 \) different sniff event lengths that are repeated in a round-robin fashion as well. The sniff event lengths have a lower bound of \( N_{\text{switch,th}} \) and an upper bound of \( 2N_{\text{switch,th}} \). As a result from the analysis, each link has the same average service time which equals to \( n/(n-1)N_{\text{switch,th}} \).

Independently from the initial credits, the switch threshold scheme always leads to a fair schedule as far as the average sniff event lengths are concerned (fairness in terms of number of slots each link gets is already provided by the basic credit scheme). Nevertheless, it may be criticized that the sniff events do not have uniform length. However, the main purpose of the switch threshold is to lengthen the sniff events and thus to reduce the number of piconet switches. This purpose is served very well since the mean sniff event length is proportional to \( N_{\text{switch,th}} \).

### 4.4. Redistribution of Credits

As stated above, the algorithm treats fully loaded links in a fair manner. However, our notion of fairness is based upon max-min fairness (cf. [8]). Following the idea of max-min fairness, a stream that does not fully use its maximum bandwidth share must release the unused bandwidth to other streams. Our algorithm does not con-
sider end-to-end streams but tries to be fair to the links a Bluetooth device has. Thus, a link that has a high amount of credits but cannot make use of it (because there is no data ready to be sent over this link) has to give (part of) its credits to the other links. Probably, the most straightforward way to perform this redistribution would be to distribute the link’s positive credit equally among the other links.

However, as already pointed out, only the difference of credits between two links is considered as being meaningful. Thus, a more suitable redistribution scheme had to be found that takes the credit difference of the links into account. Our approach calculates the amount of credits to be distributed as

\[
\frac{n-1}{n} (ac_{\text{red}} - ac_{\text{min}})
\]

with \(ac_{\text{red}}\) denoting the credits of the link to be redistributed, \(ac_{\text{min}}\) denoting the credits of the link with minimal amount of credits, and \(n\) being the number of the device’s links. This amount of credits is taken from the credit account of the link to be redistributed and is added equally to the other links.

![Figure 9: Redistribution methods](image)

After distribution, the redistributed link has approximately the same amount of credits as the link with minimum credits. In contrast to the absolute distribution of positive credits, the number of distributed credits also depends on the link that has been constrained most by the fair scheduling, i.e. the redistribution happens relatively to the link with \(ac_{\text{min}}\). Examples for both schemes can be found in Figure 9, which shows the credits of three links before and after redistribution. The credit account to redistribute has a value of 10 (\(ac_{\text{red}}\)). Therefore, the absolute redistribution scheme gives 10/2 = 5 credits to each of the two other links and subtracts the 10 credits from \(ac_{\text{red}}\). Using the relative scheme, the lowest credit value (\(ac_{\text{min}} = -8\)) is taken into account as well, yielding a higher number of credits to redistribute. In this case, the 12 credits resulting from the formula given above are transferred to the two other links. This leads to the intended situation where the redistributed and the minimal credit account both have the same amount of credits (-2) after redistribution.

Of course, the redistribution of credits takes place only if the amount of credits to be distributed is positive.

The redistribution of credits is triggered by a POLL-NUL sequence since this is a clear indication that the load on the link is lower than the share assigned to it by the scheduler. There may be other events (e.g. sniff timeout) that qualify for triggering redistribution. However, they are not considered in this paper but are subject to further research.

5. Simulation Results

In order to be able to study the properties of our approach, a Bluetooth simulation has been developed as an extension to the network simulator ns-2 [10]. Our Bluetooth simulation features a detailed model of the Bluetooth medium access control including its fast ARQ mechanism. Furthermore, it supports scatternet operation and segmentation and reassembly. As we mainly concentrate on scheduling properties, packet transmission over the wireless channel has been assumed to be error-free for the time being. Thus, there are no retransmissions because of co-channel interference between piconets. However, packets may have to be retransmitted in case of unsuccessful sniff slots, e.g. in case a master transmits a data packet while the corresponding slave visits another piconet.

5.1. Credit Scheme

We use the scenario of Figure 10 to demonstrate the fairness of our approach. It consists of three piconets. There are five CBR streams going to a “hub” device F, each using one slot packets with maximum payload. Each of the senders A through E saturates its respective link.

![Figure 10: Fairness scenario](image)
Each packet has a payload of 23 bytes (additional 4 bytes are needed for the link layer header). Because of Bluetooth’s TDD and ARQ mechanisms, there must be a send slot for each receive slot of device F. Therefore, device F is able to receive at most 800 one slot packets per second, yielding an overall rate of $800 \text{ s}^{-1} \cdot 23 \text{ Byte} \cdot 8 = 147.2 \text{ kbit/s}$. Consequently, one fifth of this throughput (29.44 kbit/s) per link is expected from a fair distribution of bandwidth. In order to avoid throughput losses that cause a deviation from the expected throughput, the piconets are synchronized to each other, i.e. there is no guard time lost after a piconet switch. Apart from using different master/slave roles for the links between F and A through E, the $T_{\text{sniff}}$ values were set to vastly different values (10, 34, 54, 112, and 164 slots, respectively) in order to be able to demonstrate the algorithm’s fairness. $T_{\text{poll}}$ was chosen sufficiently high to prevent links from reaching their poll interval. As we wanted to study the sniff event length resulting from the credit allocation scheme in this scenario, $N_{\text{switch_th}}$ was set to zero, i.e. there was no minimum service time. Although the sniff slot frequency is different from link to link, the basic credit scheme should be able to schedule the links in a fair manner (i.e. every stream should experience a throughput of about 29.44 kbit/s).

Figure 11 shows the throughput in the first 20 seconds of simulation time. It shows clearly that the bandwidth is distributed in a fair manner among the five links. In addition to that, numerical throughput values obtained from longer simulations (not depicted here) show that the expected rate of 29.44 kbit/s is actually reached. Note that this rate is reached exactly since the credit algorithm provides each link with the same number of slots in the long run.

Additionally, it is interesting to look at the distribution of the sniff event lengths in this scenario, which is depicted by Figure 12. The algorithm does not force uniform length for links with differing sniff periods. Instead, links with smaller sniff periods have shorter sniff event lengths in the average. Thus, the algorithm has the desirable property of serving links with short sniff periods more often than links with long periods in order to achieve the same bandwidth for all links.

5.2. Credit Redistribution

The scenario from Figure 10 is now reused with some modifications to illustrate the credit redistribution scheme and the algorithms adaptivity to load changes. The value of $T_{\text{sniff}}$ is set to 2 for all links; $N_{\text{switch_th}}$ is set to 50. This value was chosen in order to make the scheduler serve each link for an appropriate number of slots (about 62 slots if each link is saturated). Apart from leading to less frequent piconet switches in general, the minimum average service time has an additional benefit in relation to credit redistribution. As the minimum service time increases the service cycle length, it allows the links to collect more credits during a service cycle that can later be redistributed.

All CBR streams initially generate a load that saturates their respective link. Streams one to five are then successively reduced to 10 kbit/s at 5, 10, 15, 20, and 25 seconds respectively.

Figure 13 shows the resulting throughput without any redistribution of credits. Additionally, the sum of the throughput of all streams is depicted. At first, every stream gets the expected rate of approximately 29.44 kbit/s. The subsequent reductions of the data rate every 5 seconds clearly show that the algorithm does not reallocate the freed bandwidth to the remaining streams, as can be seen from the overall throughput sum. Instead of constantly decreasing, the sum should stay constant if the unused bandwidth were redistributed perfectly.

This decrease is due to the fact that most of the airtime allocated to the idle links is consumed by extensive polling, which does not contribute to the throughput as can be seen from the throughput sum.
polpologies as it requires no scatternet-wide coordination. Because of the low coordination overhead, it is able to rapidly adapt to changing traffic conditions.

Link level fairness is achieved through a slot accounting scheme that is able to reallocate unused bandwidth following the idea of max-min fairness. In order to increase the performance, minimum average service times were introduced. The suitability of the approach was shown by evaluating its performance using basic scatternet topologies in a Bluetooth simulation environment.

Future work will concentrate on further reducing the throughput loss due to POLL-NULL sequences by decreasing the frequency of presence points for mostly idle links. Moreover, we will evaluate more complex scatternet topologies using realistic loads. When designing the scatternet scheduling scheme, resistance against interference was already taken into account. However, its behavior under interference conditions and the appropriate amount and density of sniff slots, which is needed to achieve sufficient resilience against interference, still needs to be clarified.

References


Figure 13: Throughput without redistribution of credits

Figure 14 demonstrates the benefit of credit redistribution by depicting the throughput sums for the different redistribution methods. Compared to the case without any redistribution, the saturated streams can at least partly use the freed bandwidth when the absolute redistribution scheme is applied. Nevertheless, the method is inappropriate as it redistributes too few credits. Especially when the load of stream 4 is reduced at 20 seconds, the freed bandwidth is not reused at all and the overall throughput drops significantly. In contrast to this, the relative distribution of bandwidth performs much better. Considering the accumulated bandwidth of all streams, only little bandwidth is lost in this scenario. However, there is still bandwidth loss due to POLL-NULL sequences. The amount of overall bandwidth loss mainly depends on the service time for the links, i.e. on the value of $N_{\text{switch}}$.

Figure 14: Overall throughput sums

6. Conclusion and Further Work

This paper introduced a scheduling approach for Bluetooth scatternets that supports arbitrary scatternet topologies as it requires no scatternet-wide coordination. Because of the low coordination overhead, it is able to rapidly adapt to changing traffic conditions.