Flow-fair Intra-Piconet (FℓIP) Scheduling for Communications in Personal Area Networks[⋆]

Stefano Basagni, [#] Maurizio A. Nanni, [#] and Chiara Petrioli [*]

[#]Northeastern University, Boston, MA, 02115, U.S.A.
[*]Università di Roma "La Sapienza," Rome, Italy

***Abstract*— As a building step of designing integrated solution for forming multi-hop Personal Area Networks (PANs) of Bluetooth devices, in this paper we introduce a new intra-piconet scheduling protocol that defines the interaction among masters and slaves with an eye to the flows passing through their piconets. Differently from previous intra-piconet solutions, such as Pure Round Robin, its E-limited variation, and the more recent Credit Scheme, fairness and traffic adaptivity is now achieved on a per-flow basis. Our FLow-fair Intra-Piconet (FℓIP) scheduling identifies the flows passing through each piconet slave. Based on this information the master can efficiently manage intra-piconet resources. FℓIP is compared with the three leading solutions for intra-piconet scheduling mentioned. Through ns2-based simulations we show that FℓIP is effective in decreasing both packet drop rate as well as end-to-end packet latency.**

I. INTRODUCTION

Research on building personal area networks (PANs) of Bluetooth devices has focused on two main research topics, namely, *scatternet formation* and *intra-* and *inter-piconet scheduling*. In the first case, the problem is that of organizing the devices into the Bluetooth units of basic communications, the *piconets*. Building a piconet means electing one node as the master of the communication and choosing some of its neighbors (nodes within the master transmission range) as its slaves. Once piconets are formed, they can be joined by nodes (*gateways*) that time-share among multiple piconets to form a multi-hop networks, called *scatternet*.

The problem of scheduling the node transmissions within a piconet (intra-piconet scheduling) is vital for enabling efficient communication so that each slave gets its fair share of the piconet bandwidth. The problem of scheduling the gateways to the piconets they belong to (inter-piconet scheduling) is similarly important for enabling network-wide routing. In both cases the main issues are 1) that of being fair, i.e., allowing all nodes to be able to send their packets without incurring significant delays or having to drop packets because of buffer overflow, and 2) that of scheduling the nodes in ways that are adaptive to the intra- or inter-piconet traffic.

A general trend of the research in this field has been that of dealing with these two topics separately. A host of scatternet formation protocols have been proposed [1] as well as methods for intra- and inter-piconet scheduling [2], [3]. Recently, in [4] we tackled with the joint study of the two problems for providing an integrated solution to scatternet formation where inter-piconet communications would be efficient and swift. In particular, we chose to design formation protocol, which we called BluePleiades*, or BP* (after BluePleiades [5]), where almost all the piconets are joined by common slaves, rather than having two piconets joined through a new one formed by one slave in one piconet and one in the other. In so doing, we could use a variation of an efficient inter-piconet scheduling, called *credit scheme* [6] and evaluate the performance of inter-piconet traffic through some selected scenarios [4]. In tuning the various parameters involved in our experiments we soon realized about the importance of intra-piconet scheduling, which revealed itself as the performance bottleneck. A simple Round Robin (RR) scheme, for instance, was completely unaware of the traffic in the piconet. The master keeps *polling* nodes with no traffic to send. In our solution, therefore, we defined and implemented an intra-piconet version of the inter-piconet credit scheme, which eventually saved the day. The intra-piconet credit scheme is fair to slaves and shows some traffic adaptivity. On the other hand, it still deals with the traffic as made up of single packets. This means that in presence of packet streams (i.e., flows) this scheme is not as adaptive as needed, and it shows still quite a high packet drop rate. The reason was mainly in the fact that in order to obtain better performance, the credit scheme has to give priority to gateway nodes. Therefore, non-gateway slaves could suffer from starvation.

With this paper, we contribute to the general topic of designing integrated solutions for scatternet formation and scheduling for PANs. In particular, we introduce a new intra-piconet protocol that is fair to the slaves depending on the traffic flows that pass through them. Our FLow-fair Intra-Piconet (FℓIP) scheduling identifies the traffic flows passing through each slave, and assigns resources to the slaves on a per flow basis. Through thorough ns2-based simulations we show the effectiveness of our solution in managing intra-piconet resources. We compare FℓIP with three other intra-piconet schemes, namely, a Pure Round Robin scheme [2], a E-limited Round Robin [2] and our Credit Scheme for intra-piconet scheduling [4]. We observed that being fair and adaptive to flows, FℓIP obtains lower drop rates in all the investigated scenarios,

---

[⋆] This paper was supported in part by NSF grant CCF 0634847 and by the International FIRB RBIN047MH9.

and assigns piconet resources significantly better than the other schemes.

The paper is organized as follows. In the next section we introduce FℓIP in detail. Section III illustrates the outcome of our comparative performance evaluation. Finally, Section IV concludes our paper.

II. FLOW-FAIR INTRA-PICONET SCHEDULING

The intra-piconet polling scheme presented here aims at being strongly adaptive to variation in traffic, especially to packet flows that come and go through a piconet, and at enabling fairness among the piconet slaves. In order to do so, our mechanism seeks fairness on a per flow basis: Resources (e.g., bandwidth) are fairly distributed among the flows detected on the links of the piconet. Here is how this is done.

For each of its slaves the master maintains a number of pebbles. Each slave has initially the same number of pebbles. The master also maintains a "bucket," which contains the pebbles consumed by the slaves. Every time the master polls a slave, a pebble is moved from that slave bunch to the bucket.

Slaves are polled based on the pebbles they have: Each time, the master polls the slave with the tallest bunch (i.e., the highest number of pebbles). Over this basic mechanism [4] we define ways for 1) what to do with a slave's bunch when a POLL/NULL occurs,[1] and 2) how to re-assign pebbles from the bucket back to the slaves bunches. In the first case, the slave bunch is decreased to equate the minimum value among all bunches. The number of pebbles subtracted is moved to the bucket. The slave is also deprived of a further constant number k of pebbles, which are also moved to the bucket. The reason of this last penalty is because when traffic is balanced on the links of the piconet, and hence the number of pebbles are basically the same for each slave, a POLL/NULL slave could be polled again soon, since the minimum number of pebbles is very close to the maximum. This mechanism enables traffic adaptivity, since nodes that have no more traffic (POLL/NULL slaves) are not polled again for a certain time.

Flow fairness is achieved by the following scheme for re-distributing the pebbles back from the bucket to the slave bunches. Based on the current number of slaves and of flows in the piconet links the master keeps computing the following value:

$$t = \sum_{s \in S} (1 + \max(f_{in}(s), f_{out}(s)),$$

where S is the current set of slaves and $f_{in/out}(s)$ is the number of current flows from/to s to/from the master. Every time the bucket has t pebbles each slave s gets back $1 + \max(f_{in}(s), f_{out}(s))$ of them. Fairness to the slaves is achieved because each one of them gets at least a pebble. Flow fairness and adaptivity is instead enforced by giving more pebbles to those slaves that have currently more flows traversing them.

What is left to explain is how the master recognizes a new flow and how it realizes that a flow expires. A traffic flow can be represented as a pair (s, d) where s and d are any two nodes in the network, the flow source and its destination. Information about s and d can easily be obtained from the transport packet header, as soon as a whole packet has been received by the transport layer. The master maintains a *flow table* that lists all the active flows. For each flow the table records the source, the destination and the time the last packet from that flow has been received. When the master receives a transport packet with a new pair (s, d) it adds a corresponding new record to its *flow table*. As mentioned, upon receiving a packet from a flow that is already in the table a master updates the time in the corresponding entry. The termination of a flow is determined by checking whether a given time has passed since a packet has been received from that flow.

III. EXPERIMENTAL RESULTS

FℓIP has been tested via ns2-based simulations to demonstrate its effectiveness in providing traffic adaptive and fair intra-piconet scheduling. Experiments are also provided to show the performance improvements obtained by using FℓIP with respect to different intra-piconet scheduling protocols for end-to-end scatternet flows.

All intra- and inter-piconet scheduling schemes have been implemented by using *BlueBrick* [4]. We have performed two sets of simulations. The first set concerns a comparative performance evaluation of FℓIP vs. three other intra-piconet schemes, namely, the Pure Round Robin polling scheme [2], the E-limited Round Robin [2] and the intra-piconet Credit Scheme first defined in [4]. The Pure Round Robin scheme is here evaluated as a base performance case, to show the remarkable improvement that can be obtained via adaptive schemes. The E-limited Round Robin has been shown to outperform Pure Round Robin as well as Exhaustive Round Robin [7]. (In our simulations, based on the selected scenarios, we have fine-tuned the value for E to 20.) The intra-piconet Credit Scheme is a traffic adaptive polling scheduling that has been proven to outperform other intra-piconet scheduling solutions, especially in an unbalanced traffic scenario. It is also an example of a polling scheme explicitly defined to work in conjunction with inter-piconet scheduling. The experiments in this first set concern one-piconet scenarios, with different types of traffic flows and different number of nodes per piconets.

The second set of experiments concerns the implementation and testing of FℓIP in scatternets where data flows are passing through piconets. These experiments are aimed at showing how the advantages of FℓIP within a piconet correspond to global improvements in terms of inter-piconet, and hence of scatternet performance.

---

[1] A single master-to-slave/save-to-master pair is called a *frame*. A POLL/ NULL frame occurs when both master and slave have no packets to transmit to each other. POLL/NULLs should obviously be avoided, since they are the main way to waste bandwidth. A good intra-piconet scheme should maximize "data/data" frames, where both the master and the slave transmit data.

All our results achieve a statistical confidence of 95% and a precision of 5%.

A. One-piconet experiments. In this set of experiments we consider piconets with a variable number of slaves s, namely, 5 and 7. Traffic flows are generated according to a Poisson arrival process with average λ. We consider values of λ so that the drop rate observed never exceeds 25% of the total number of packets generated. Each flow is assigned to a random source/destination pair. Whenever a flow is established packets are generated at a given data rate. Each packet is 200B long. Packets are fragmented by the node L2CAP layer into 8 ACL DH1 packets. The buffer dimension of each node is limited to 100 packets.

We consider the following metrics: 1) The packet drop rate, and 2) the end-to-end packet latency. Each simulation lasts $(10+300)$s. The first 10s are needed to create the piconet.

A1) 25Kbps flow data rate; 0.5s flow duration. The first set of experiments concerns piconets with 5 and 7 slaves, where flows that last 0.5s each are generated at the rate of 25Kbps. The traffic is uniformly distributed to the nodes.

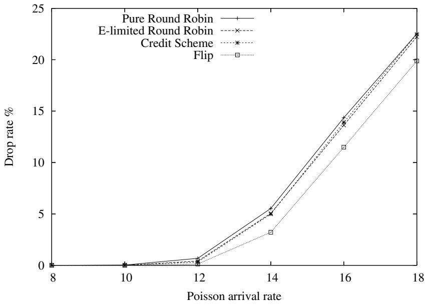

Fig. 1. Drop rate in a 5-slave piconet

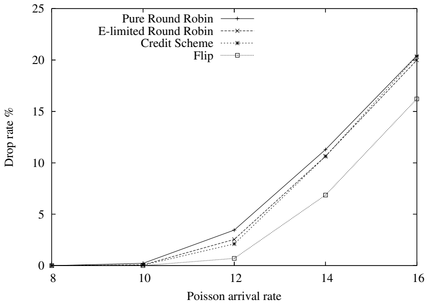

Fig. 2. Drop rate in a 7-slave piconet

As expected, the drop rate grows with increasing number of flows. As depicted in Fig. 1 and Fig. 2 the two round robin schemes and the Credit Scheme show comparable performance. This result is no surprise: Being the traffic quite balanced the two forms or round robin act basically in the same way. A similar reasoning applies to the Credit Scheme. Credits are uniformly distributed to all slaves, and they are removed from a slave based on the occurrence of a POLL/NULL. Being the traffic balanced, and being the number of POLL/NULLs basically the same for each slave, they are polled with the same frequency, and therefore the drop rate is comparable.

FℓIP shows a lower drop rate. The reason is twofold. First, we observe that FℓIP is successful in decreasing the number of POLL/NULLs. This is due to the per-flow way with which FℓIP assigns resources, i.e., pebbles, to each slave. The higher the number of flows, the more the pebbles, the more the master keeps staying with a slave, instead of polling some other less loaded slave. In numbers: FℓIP delivers about 6% more packets than Pure Round Robin and about 5% more packets than both E-Limited Round Robin and Credit Scheme when the traffic arrival is the highest. Furthermore, FℓIP is effective in increasing the number of frames where data are sent both from the master and from the slave. Since the number of re-assigned pebbles depends on $\max(f_{in}(s), f_{out}(s))$, each pebble pays for one in-flow and one out-flow. This increases the number of data/data frames. 90% of the FℓIP frames are data/data frame, vs. 83% of the other schemes.

From the two figures above we also see that the performance of all protocols is better for piconets with 5 rather than 7 slaves. This might sound surprising, given that we generate the same amount of traffic in both cases. However, we notice that increasing the number of slaves means decreasing the amount of traffic generated by each of them. Therefore, the master serves a bigger number of slaves with a smaller amount of traffic per slave. In the 7 slave scenario there is a higher probability that there are no packets to exchange in the up-link and/or in the down-link (if both, master and slave will perform a POLL/NULL sequence). In other words there is a higher probability that some of the slots will not be used to transmit data packets, which implies worse performance in bigger piconets. In this toughest setting FℓIP is effective in reducing not only POLL/NULL frames, but also POLL/data and data/NULL frames, which are highly likely to occur in these scenarios.

The FℓIP end-to-end packet latency is also always shorter than that of the other solutions. The actual values for the latency are very close for all four schemes. However, we compute the end-to-end latency only for the packets that are actually delivered to their intended destination. Therefore, within the same delay, FℓIP is able to deliver a higher number of packets.

Given the similar trends in both 5 slave and 7 slave piconets, in what follows we show results only for the latter case.

A2) 25Kbps flow data rate; 3s flow duration. In this second experiment, we consider 25Kbps flows, each lasting six times more than before, i.e., 3s. In this scenario, the traffic is more unbalanced than for the first set of experiments. We generate up to 2 flows per second, which implies a higher probability that some slaves will not generate traffic at all for some time. Fig. 3 depicts the four protocol performance concerning the drop rate. Non-traffic adaptive solutions, like the Pure Round Robin, suffer from the more unbalanced traffic. The E-limited Round Robin and the Credit Scheme show better performance than the pure Round Robin. FℓIP delivers up to 12% more packets than

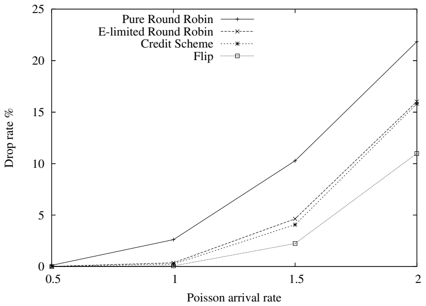

Fig. 3. Drop rate in a 7-slave piconet

Pure Round Robin, and up to 7% more packets than both E-Limited Round Robin and Credit Scheme.

Finally, E-Limited Round Robin, Credit Scheme and FℓIP show a much lower end-to-end packet latency than Pure Round Robin. We observed an average of 1.4s higher latency for packets delivered by this scheme. This proves once more the Pure Round Robin cannot efficiently deal with unbalanced traffic. A strongly traffic adaptive algorithm such as FℓIP performs better both for what concerns drop rate and end-to-end packet latency.

We have also looked at scenarios of piconets with 5 and 7 slaves generating 25Kbps traffic each lasting 60s. In these scenarios the traffic is completely unbalanced. As expected the Pure Round Robin scheme shows quite an impressive drop rate (40% of the packets) and a very high end-to-end packet latency (an average of 5s). What is interesting to notice in this last batch of experiments is that differently from before, most of the packets are dropped at the source node, rather than at the master. Never leaving their source, these packet do not steal bandwidth to other packets. This justifies the observed end-to-end latency, which is slightly higher in those algorithms that have the lowest drop rate. FℓIP imposes delays that are slightly higher of the E-limited Round Robin and of the Credit Scheme: An average of 1.66s vs. 1.45s and 1.35s, respectively (where the drop rate is 11% for FℓIP, 15% for the other two schemes).

B. Scatternet experiments. In this set of experiments we show how the four protocols perform when deployed in a scatternet. In our simulations we generated 100 different 30-node BP* [4] connected scatternets, where the nodes are initially scattered on a square are of 900m^2 area. The inter-piconet scheduling algorithm we implemented is the enhanced version of the inter-piconet Credit Scheme described in our previous work [4]. Routing is shortest path and the traffic is generated as before (we consider flows lasting 0.5s). For each flow, source and destination nodes are chosen randomly among the 30 nodes in the scatternets.

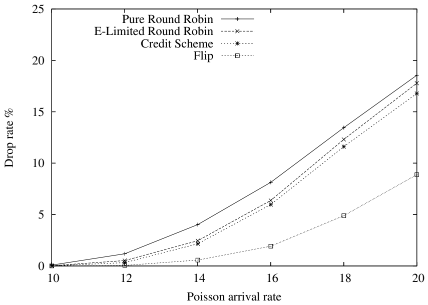

Fig. 4. Drop rate (%) in a 30-node scatternets

Our results show once more the effectiveness of FℓIP in handling the flow traffic through a piconet via dealing appropriately with the gateways. FℓIP outperforms the other schemes for both drop rate and end-to-end packet latency. Fig. 4 shows that FℓIP incurs 10% less dropped packets than the the Pure Round Robin scheme and 8% less losses than both E-limited Round Robin and the Credit Scheme. Although less packets are dropped, FℓIP is still faster that any other scheme in delivering packets to their destination: 50% faster than the Pure Round Robin and 15% faster than the other two schemes.

IV. CONCLUSIONS

In this paper we presented a new intra-piconet scheduling algorithm for Bluetooth scatternet that achieves fairness and traffic adaptivity. Differently from previous solution, fairness and adaptivity is on a per-flow basis. We have compared our protocol, termed FℓIP, with three other leading solutions for intra-piconet scheduling, namely, Pure Round Robin, E-limited Round Robin and the more recent Credit Scheme. We observed that, for what concerns relevant metrics such as packet drop rate and end-to-end latency, FℓIP outperforms all the other schemes in all scenarios.

REFERENCES

[1] I. Stojmenovic and N. Zaguia. Bluetooth scatternet formation in ad hoc wireless networks. In J. Mišić and V. B. Mišić, editors, *Performance Modeling and Analysis of Bluetooth Networks: Network Formation, Polling, Scheduling, and Traffic Control*, chapter 9, pages 149–171. CRC Press, Boca Raton, FL, 2006.

[2] S. Capone, R. Kapoor, and M. Gerla. Efficient polling schemes for Bluetooth picocells. In *Proc. of IEEE International Conference on Communications ICC 2001*, volume 7, pages 1990–1994, Helsinki, Finland, April 2001.

[3] J. Mišić and V. B. Mišić. Intra-piconet polling algorithms in bluetooth. In A. Boukerche, editor, *Handbook of Algorithms for Wireless Networking and Mobile Computing*, chapter 31, pages 725–734. Chapman & Hall/CRC Press, Boca Raton, FL, 2005.

[4] S. Basagni, M. A. Nanni, and C. Petrioli. Bluetooth scatternet formation and scheduling: An integrated solution. In *Proceedings of IEEE MILCOM 2006*, pages 1–7, Washington, DC, October 23–25 2006.

[5] D. Dubhashi, O. Häggström, G. Mambrini, A. Panconesi, and C. Petrioli. Blue pleiades, a new solution for device discovery and scatternet formation in multi-hop bluetooth networks. *ACM/Springer Wireless Networks*, 13(1):107–125, January 2007.

[6] S. Baatz, M. Frank, C. Kuhl, P. Martini, and C. Scholz. Bluetooth scatternets: An enhanced adaptive scheduling scheme. In *Proceedings of IEEE Infocom 2002*, volume 2, pages 782–790, New York, NY, June 23–27 2002.

[7] J. Misic, K. L. Chan, and V. B. Misic. Performance of Bluetooth piconets under e-limited scheduling. In *Tech. report TR 03/03*, Department of Computer Science, University of Manitoba, Winnipeg, Manitoba, Canada, May 2003.