BlueFlows: Routing and Flow Admission in Bluetooth PANs*

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

ECE Dept., Northeastern University, Boston, MA 02115, {basagni,mnanni}@ece.neu.edu
* Dipart. di Informatica, Università di Roma "La Sapienza," Roma, Italy, petrioli@di.uniroma1.it

***Abstract*— This paper concerns routing and flow admission controls in personal area networks (PANs) of Bluetooth devices. We define a solution, termed *BlueFlows*, that jointly implements flow routing and admission. BlueFlows guarantees 0% delivery of all packets of every admitted flow, while achieving low end-to-end latencies and inter-packet delay. Through ns2-based simulation we compare the performance of our solution to that of a shortest path flow routing (SP) with no admission control. In scenarios with medium/high traffic BlueFlows admits remarkably more flows than SP and delivers packets with consistently lower end-to-end latency and inter-packet delays.**

I. Introduction

Since its release in the late nineties Bluetooth (BT) [9] has become the leading wireless technology for short range communications. Along with the host of research and commercial products on the *single-hop* applications of this technology (mostly cable replacement), research has been pursued to explore the BT provision for *multi-hop* communications. In particular, by forming a *piconet* with one master and multiple slaves, devices can communicate directly, and, by time-sharing one or more of the piconet members, ways are provided for interconnecting multiple piconets into a *scatternet*, i.e., a multi-hop network of BT devices. Many works have been published on scatternet formation protocols [1], and on the scheduling that governs master/slave communications (*intra-piconet scheduling* [2]) as well as that concerning the time sharing of devices among piconets (*inter-piconet* scheduling [3]). Furthermore, BT routing protocols have been presented that take into account the unique characteristics of these networks. Most of the solutions available for intra- and inter-piconet scheduling and for routing 1) solve these problems for packet-based communications, where schedules and routes are found for single packets, and 2) deal with these problems separately, i.e., without a uniform approach to the design of scheduling and routing as a whole. The few previous solutions that tackle the problem of routing *flows* (ordered sequence of packets from a source to a destination) rather than single packets are confined to very specific topologies and routing [4] or small networks [5], and deal with flow admission without jointly considering intra- and inter-piconet scheduling design for overall performance optimization.

In this paper we propose *BlueFlows*, a unified mechanism for flow-based communications in multi-hop scatternets where *flow routing* and *flow admission control* act together to enable end-to-end efficient communications in BT scatternet. Differently from leading solutions for multi-hop routing BlueFlows is not based on flooding, thus avoiding the excessive overhead of route discovery that those protocols entail. Furthermore, BlueFlows is able to admit and route flows guaranteeing 1 0% packet delivery ratio and achieving low packet latency and inter-packet delay at destination. Through thorough ns2/BlueBrick [6]-based simulations we have evaluated the performance of BlueFlows and compared it to that of shortest path routing (SP) with no flow admission control. In the considered scenarios, we have observed that as soon as traffic requirements become demanding BlueFlows admits up to six times the number of flows with 1 0% delivery admitted by SP, while keeping the average end-to-end packet delay to no more than 7% than that of SP. Finally, the inter-packet delay at destination—a metric important for many flow-based applications—induced by BlueFlows is independent of the traffic load, and, at high traffic, is half of that of SP packets.

The rest of the paper is organized as follows. Section II describes BlueFlows. In Section III we show the results of our simulation-based comparative performance evaluation.

Finally, Section IV concludes the paper.

II. BlueFlows

Routing and flow admission control are tightly coupled in BlueFlows, in the sense that if a *feasible route* is not available the flow is not admitted. A node that generates a flow starts an exhaustive route search to the destination that proceeds through neighboring nodes based on the resources they have for admitting that flow. The route discovery packets (RDPs) are not flooded through the network. Rather, the generic node that can support the flow selects a next hop neighbor to which it forwards the RDP, and locks the necessary resource to admit that flow. This process proceeds recursively through the network until either the destination is found or a node is reached that cannot admit the flow. In the first case a positive Routing Response Packet (RRP+) is sent back to the source confirming to nodes in the route with locked resource that a flow is coming through them. In the second case a negative RRP (RRP-) is sent back to the node in the path that has some other neighbors to send the RDP packet to. Nodes that propagate back the RRP- made the locked resources available again for flow admission. A node decides whether

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

to admit a flow or not depending on its *current load*, which in turn depends on the set of flows currently handled by the node and on its *maximum load*. The latter is defined as the maximum number of flows that a node can handle without dropping packet. Computing a node's maximum load is one of the critical operations of BlueFlows, and it is described in details below. The current load of a node is the maximum between the number of its incoming flows and the number of its outgoing flows.

The following is a description of BlueFlows. We consider all flows having the same priority. Furthermore, given a node n and a flow f, a *feasible next hop* of n is a neighbor n' such that n' allows f's packets to reach their final destination and such that routing f through n' does not make n's current load exceed its maximum load.

1. When a node s needs to start a traffic flow toward a destination node d, it creates a Routing Discovery Packet (RDP) and sends it to a feasible next hop. If multiple feasible next hops are available, those that reach the destination with the smallest number of hops are preferred. Possible ties are broken by choosing as next hop the node that is handling the smallest number of flows (i.e., s tries to balance the load over its links), and eventually by using the nodal unique ID. If no feasible next hop is found, the new flow is discarded.

2. When a node n receives an RDP packet, it first verifies that the new traffic flow can be admitted as an IN flow. To this aim, n computes its current load and checks whether by accepting the new IN flow it exceeds its maximum load or not. If it does, n sends back a negative Routing Response Packet (RRP-). Otherwise, it looks for a feasible next hop as described in Step 1. If such neighbors can be found, n adds the new flow to its flow table and forward the RDP packet to the selected feasible next hop. Resources (bandwidth, buffer space, etc.) are reserved for this flow until an RRP- packet is received. If no feasible next hop is available n sends back an RRP-.

3. When the destination node d receives an RDP packet, it checks whether the flow can be admitted. If so, a feasible route for the flow has been discovered. In this case, node d sends back an RRP+. Every node on the feasible route forwards back this packet to node s and records that the flow has been admitted. If d cannot admit the flow the packet propagated back to s is an RRP-.

4. When a node n receives an RRP- packet from the node it had chosen as next hop node, it looks for a new feasible next hop to get to the destination. If a new feasible next hop is found, n sends to it a new RDP packet. If no new feasible next hop is found, n sends back an RRP- packet to the previous hop. If $n = s$ the flow is rejected.

A key parameter for BlueFlows is a node's maximum load. In general, given a node n, we can compute the maximum theoretical uplink/downlink throughput, which is obtained when n transmits a packet in each slot. For instance, using the latest BT specifications 3-DH1 packets (each requiring one time slot) with a payload of 81B we have that the maximum theoretical throughput per second attainable for the uplink (downlink) is $T = 81 * 8 * 1\,60\,/2 = 5\,8$ .4Kbps, where $1\,60$ is the number of BT slots in a second. Let us consider applications that generate 0 B data packets at 3 Kbps, i.e., that generate 7.81 packets per second. The BT L2CAP level fragments every packet into $\lceil \frac{0}{81} \rceil = 5$ BT 3-DH1 packets (4B are needed for the L2CAP header). If all data flows are 3 Kbps each node cannot admit more than 0 flows, with a corresponding throughput of 0 Kbps (which is close to the theoretical maximum). However, this upper bound for a node maximum admissible load does not take into account the non-negligible amount of control packets that go into intra-piconet and inter-piconet coordination. In particular, we should also consider the bandwidth wasted because of interference, routing control packets, inefficiencies that are inherent to the intra- and inter-piconet scheduling algorithms as well as the role of the node. In what follows we briefly describe how we actually computed the maximum load that a node can handle depending on whether it is a master, a simple slave, or a gateway. (These are the actual values used in our experiments below.) In the discussion that follows we consider *FℓIP* as the intra-piconet scheduling used in our scatternets [7], and a flow-based version of the credit scheme of [8] that we developed where credit distribution is based on the number of flows currently active in a piconet.

1. Master node. Let m be the master of a piconet. The bandwidth estimation B_m for the uplink (downlink) is given by the theoretical upper bound T minus the number of slots w_m that m wastes polling gateways that are not in its piconet, the number of slots c_m spent by m to synchronize with the gateways that are in its piconets, and the number of slots needed by routing control messages (r_m) and those wasted because of interferences (i_m). In other words: $B_m = T - 81 * 8(w_m + c_m + r_m + i_m)$. In order to compute w_m we need to know the probability that a gateway actually switches to m's piconet, which can be computed by knowing the inter-piconet scheduling and the length of the *sniff* period that implements it as follows:

$$w_m = \sum_{i=1}^{k} \frac{1\,60}{t_{sniff}} * \left[1 - \frac{1}{n_i}\right] * (t_{sniff} \quad _{tm} \quad + 2 .$$

The same parameters are used to compute c_m for the uplink (downlink):

$$c_m = \sum_{i=1}^{k} \frac{1\,60}{t_{sniff}} * \frac{1}{n_i}.$$

The mechanism of master-gateway synchronization allows both to temporarily extend the sniff attempt exactly for the time the gateway stays in the piconet. Without this extension, if the master does not address the gateway for t_{sniff} $_{tm}$ [9] slots, their link goes to low-power mode and the gateway stops listening to the master and waits for the next presence point with another piconet. The (average) number r_m of packets exchanged each second by master m for route construction depends on the number of flow requests. As such, it can be estimated at run time by every node based on historical data. Finally, m can also

easily estimate the number i_m of slots wasted each second because of interferences.

2. Slave node. Let s be a slave in the piconet of master m. A tighter upper bound on the number of flows that s can admit can be computed by estimating the bandwidth B_s in the worst case when all the flows in the piconet generate or terminate at s. In this case, B_s for the uplink (downlink) is given by the theoretical upper bound T minus the number of slots that m does not give to s, i.e., the pn_m slots wasted for polling slaves that will reply with a NULL packet, the w_m and c_m slots wasted by m to poll gateways and the r_m slots needed by routing control messages. (Unfortunately, a slave cannot estimate the number i_m of slots that are gone in a second because of interferences.) More precisely: $B_s = T - 81*8(pn_m + w_m + c_m + r_m)$. While computing w_m, c_m and r_m is similar to the master case, the number of POLL/NULL per second wasted by the master for polling other slaves can be computed by knowing the flow-based credit distribution mechanism for the intra-piconet scheduling FℓIP. In particular, pn_m depends on the number of slaves ns, on the maximum number f of flows in or out of the nodes in the piconet, and on the number $j \geq 0$ of credits that FℓIP subtracts from slaves each time it resets an account [7]:

$$pn_m = \frac{80 \quad * ns}{\left(\left\lfloor \frac{(\text{s} \quad -1)j}{\text{s} \quad +f} \right\rfloor f + j\right)(ns + f)}.$$

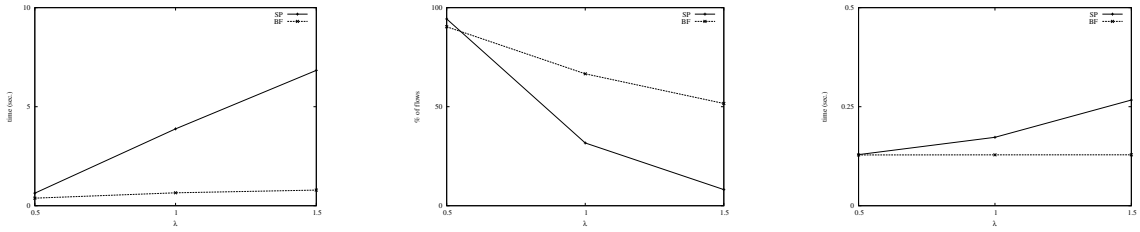
3. Gateway node. Estimating the effective bandwidth B_g that a gateway g can effectively use, and therefore the maximum number of flows it can accept is even more difficult than for a regular slave. As in the case of a regular slave, we can estimate the slots per second that are spent for routing control packets. As in the case of a regular slave, g cannot estimate the number of slots that are wasted because of interferences. However, inter-piconet scheduling plays an important role on taking further slots away each second because of the mechanism provided by these scheduling protocol for efficient handling of unbalanced traffic scenarios. In particular, in the flow-based intra-piconet scheduling that we use for our experiments, adaptiveness to variation in traffic is obtained by having gateway g leaving the piconet after a POLL/NULL. From the time when the POLL/NULL occurs to the next presence points for g the gateway is idle, i.e., all those slots are wasted. In order to compute B_g we consider the status of its queues. In particular, g keeps an history of the queue status and periodically, based on this history, it updates the possible maximum load it can support, i.e., the maximum number of flows it can admit. (A detailed description on how to compute B_m, B_s and B_g will be given in the extended version of the paper.)

III. Simulation Results

In order to assess the effectiveness of BlueFlows in admitting flows and never drop their packets, we compare its performance versus that of a shortest path (SP) routing with no flow admission control, i.e., where all flows are admitted and their packets routed through the shortest path from their source to the intended destination. Both protocols have been implemented in *BlueBrick*, the BT extension to the ns2 simulator that we developed in [6]. Both protocols, BlueFlows and SP, use the flow-based intra-piconet scheduling protocol FℓIP [7] and a flow-oriented inter-piconet scheduling protocol we developed on purpose, based on the Credit Scheme of Baatz et al. [8]. Scatternets are generated according to the scatternet formation protocol Blue Pleiades [10] executed by N BT nodes scattered uniformly and randomly in a square area of 90 m^2. We investigated networks with different densities, as generated by values of N from the set $\{\text{0} \ , \text{0} \ , 70\}$. (Because of lack of space, we show here only results obtained for networks with 0 nodes. Results for other densities show similar trends.) Traffic flows are generated according to a stochastic Poisson arrival process with parameter λ (average number of flows generated per second). We selected three values for λ, which account for medium/high traffic loads. Once a traffic flow is generated, its source and a destination nodes are chosen randomly and uniformly among the nodes in the scatternet. Every flow lasts 60s and generate 0 B UDP packets at the fixed data rate of 2 Kbps, which corresponds to 7.81 UDP packets per second. As mentioned, every UDP packet is fragmented by the BT L2CAP layer into BT 3-DH1 packets as described in the Bluetooth 2.1 specifications, after adding a 4 bytes L2CAP header. A 3-DH1 packet carries a 81B payload and is transmitted in one time slot. Therefore, a 0 bytes UDP packet (0+ 4 after the L2CAP layer has processed it) requires 5 BT 3-DH1 fragments (0 bytes of total payload) to be transmitted. (We carefully choose this UDP size in order to minimize the wasted payload in the last 3-DH1 fragment.) When a flow is generated, a flow request is sent to the network layer of the BT stack. Based on the routing algorithm, the network layer will perform some actions in order to decide whether to admit the flow or not. If the traffic flow has been accepted by the network layer, the application starts to send the UDP packets to the L2CAP layer; Otherwise, the flow is dropped. The metrics investigated in this paper concern *end-to-end packet latency*, the *flow admission ratio*, and the *inter-packet delay* between consecutive packets of the same flow measured at destination. Each experiment lasts one hour of simulated time. Values are collected at steady state, which happens after about 1 0 s. All the results we show are obtained by average over 1 0 different topologies (9% confidence and % precision).

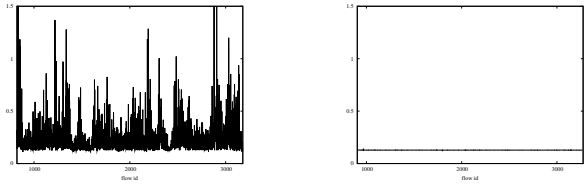
Fig. 1(a) depicts the average end-to-end latency experienced by the packets of all flows. With no flow admission, as soon as the flow requests become demanding network congestion builds up that imposes an average of 7s packet travel time from source to destination. BlueFlows has been designed instead to admit only the traffic that the nodes can handle. As a result, in networks with BlueFlows routes that average 4.3 hops (1 % longer than SP routes), latency averages at 0.5s.

Fig. 1(b) depicts the percentage of flows that are able



(a) End-to-end packet latency (b) Percentage of flows with 1 0% delivery (c) Inter-packet delay over all flows

Fig. 1. Investigated metrics: Average latency, delivery ratio and average inter-packet delay



(a) SP (b) BlueFlows

Fig. 2. Average inter-packet delay per flow

to deliver all the flow packets. Since BlueFlows delivers all packets, its curve shows the percentage of flows that are admitted in the network among all those generated. We observe that at high traffic loads, the number of admitted flows with 0% drop rate is much higher by using BlueFlows than SP. The case with medium traffic shows a slightly better performance of the strategy that accepts all generated flows. This is because for achieving 0% packet drop, BlueFlows needs to be conservative in the estimation of the maximum load. Furthermore, since BlueFlows locks the resources in the intermediate nodes until a RRP- packet is back, there are times in which part of the bandwidth is reserved and not used. In situation of very high traffic, our solution allows the network to handle correctly more than 0% of the generated flows. In the same case, SP meets the delivery requirement only for 8% of the generated flows. In Fig. 1(c) we show results for the inter-packet delay between two consecutive packets of the same flow. Since every flow in our simulation is a 2 Kbps flow and each packet is 0 B, we generate 7.81 packets per second, i.e., the inter-packet delay of packets from the application is 0.1 8 s. BlueFlows practically allows the destination to receive the flow packets with the same inter-packet delay (on average), independently of the traffic load. For SP, instead, this metrics is tightly dependent on the λ. In fact, not only the average lag between two consecutive SP packets over all flows grows to over twice that of the application, but the actual average inter-packet delay can be as high as 1.5s. This is shown in Fig. 2 where for each flow at steady state we show its own average inter-packet delay. We observe that while SP averages are remarkably varied (Fig. 2(a)), BlueFlows obtains the same average inter-packet delays at the source (from the application) and at the destination for all traffic flows that are admitted (Fig. 2(b)).

IV. Conclusions

In this paper we have investigated routing and flow admission controls Bluetooth PANs. Our solution, termed *BlueFlows*, takes care of both flow routing and admission. Through ns2-based simulations we showed that BlueFlows guarantees the 1 0% delivery of all packets for every admitted flow, while achieving low end-to-end latencies and inter-packet delay. In particular, we observed that in scenarios with medium/high traffic BlueFlows admits remarkably more flows than shortest path routing with no admission control, and delivers packets with consistently lower end-to-end latency and inter-packet delays.

References

[1] S. Basagni, R. Bruno, and C. Petrioli, "Scatternet formation in bluetooth networks," in *Mobile Ad Hoc Networking*, S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic, Eds. Piscataway, NJ and Hoboken, NJ: IEEE Press and John Wiley & Sons, Inc., April 2004, ch. 4, pp. 117–137.

[2] J. Mišić and V. B. Mišić, "Intra-piconet polling algorithms in-Bluetooth," in *Handbook of Algorithms for Mobile and Wireless Networking and Computing*, A. Boukerche, Ed. Boca Raton, FL: Chapman and Hall/CRC Computer and Information Science, November 28 2005, ch. 31, pp. 725–734.

[3] ——, "Bridges of Bluetooth county: Topologies, scheduling and performance," *IEEE Journal of Selected Areas in Communications, Wireless Series. Special issue on Wireless LANs and Home Networks*, vol. 21, no. 2, pp. 240–258, February 2003.

[4] A. Zanella, D. Miorandi, S. Pupolin, and P. Raimondi, "On providing soft-QoS in wireless ad hoc networks," in *Proceedings of WPMC 2003*, Yokosuka, Kanagawa, Japan, October 21–22 2003.

[5] Y.-S. Chen and T.-H. Lin, "A time-slot leasing-based QoS routing protocol over Bluetooth WPANs," *International Journal of Ad Hoc and Ubiquitous Computing, Special Issue on Pervasive Computing Through Networked Sensing Devices, Lai et al., eds.*, vol. 2, no. 1/2, pp. 92–108, 2007.

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

[7] ——, "Flow-fair intra-piconet (FℓIP) scheduling for communications in personal area networks," in *Proceedings of the IEEE Radio and Wireless Symposium, RWS 2008*, Orlando, FL, January 22–25 2008, pp. 839–842.

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

[9] http://www.bluetooth.com, *Specification of the Bluetooth System, Volume 1, Core*. Version 2.0 + EDR, November 8 2004.

[10] 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*, vol. 13, no. 1, pp. 107–125, January 2007.