A Context and Content-Based Routing Protocol for Mobile Sensor Networks*

Gianpaolo Cugola and Matteo Migliavacca

Dipartimento di Elettronica e Informazione, Politecnico di Milano, Italy
[cugola,migliava]@elet.polimi.it

Abstract. The need of monitoring people, animals, and things in general, brings to consider *mobile WSNs* besides traditional, fixed ones. Moreover, several advanced scenarios, like those including actuators, involve *multiple sinks*. Mobility and multiple sinks radically changes the way routing is performed, while the peculiarities of WSNs make it difficult to reuse protocols designed for other types of mobile networks.
In this paper, we describe CCBR, a *Context and Content-Based Routing* protocol explicitly designed for multi-sink, mobile WSNs. CCBR adopts content-based addressing to effectively support the data-centric communication paradigm usually adopted by WSN applications. It also takes into account the characteristics (i.e., context) of the sensors to filter data. Simulations show that CCBR outperforms alternative approaches in the multi-sink, mobile scenarios it was designed for, while providing good performance in more traditional (fixed) scenarios.

1 Introduction

The recent advances in WSNs are rapidly expanding the range of applications they can support: from "traditional" environmental monitoring, where a number of nodes is scattered across an area collecting data for a single sink, to *mobile scenarios* involving *multiple sinks*. This happens when the entities to monitor are animals (e.g., in farming scenarios), people (e.g., in elderly care scenarios), or things moving around (e.g., in logistics), while several mobile devices (e.g., PDAs) are used as sinks, or actuators, also acting as sinks, are involved.

Unfortunately, mobility and the presence of multiple sinks is something that has been largely neglected by research on WSNs so far, especially if we consider the case of data-aware routing protocols. Indeed, one of the main peculiarities of WSNs is the data-centric form of communication that they usually adopt: a few *sinks* (a single one in the simplest scenarios) are interested in receiving only some specific data among those collected by sensors, e.g., the temperature readings that exceed some threshold. This suggests abandoning traditional, address-based routing protocols, to adopt a *Content-Based Routing* (CBR) [1] protocol, in which messages do not carry any explicit address, while they are routed based

* This work has been partially funded by the European Community under the IST-034963 WASP project and by the Italian Government under the MIUR-FIRB program INSYEME.

on their content and on the interests specified by nodes. This is the solution taken by popular protocols for WSNs like Directed Diffusion [2] and TinyCOPS [3], without however considering mobility as a key aspect.

Moreover, if we look at the typical scenarios of usage of a WSN, we may notice that communication is not only data-centric but it is also often *context-aware*. As an example, a farmer could be interested in knowing the activity level of "young" cattle only, while, in a logistics application, different temperature thresholds are critical for different goods. Encoding such context-awareness as part of the message content and using standard CBR to route messages is possible, but can be inefficient, increasing message size and communication overhead.

Starting from these considerations, we developed *CCBR*, a *Context and Content-Based Routing* protocol for *multi-sink, mobile WSNs*. It adopts a probabilistic, receiver-based approach to routing, where each node decides autonomously if forwarding packets, loosely collaborating with others in keeping routing information up-to-date. This approach has proven to be well suited to support the multi-sink, mobile scenarios we target, also being able to efficiently address the (albeit slower) dynamics inherent in traditional, non-mobile WSNs.

The next section goes into the details of the protocol, explaining how it operates, while Sect. 3 evaluates the performance of CCBR through simulation, comparing it with other approaches. Finally, Sect. 4 surveys related work and Sect. 5 draws some conclusions and describes our future plans.

2 Context and Content-Based Routing

The reference scenario for CCBR is that of a WSN composed of a set of *nodes* possibly moving around at different speeds. Among these nodes we distinguish between those that are interested in receiving messages (the *sinks*), and those that send out messages (the *sensors*). In such a network: (i) the same node may act both as a sink and as a sensor, and (ii) messages do not necessarily carry sensors readings. As an example, a node could collect temperature readings from other nodes (acting as a "sink") to notify fire alarms (acting as a "sensor").

2.1 The API

CCBR offers three main primitives to the upper layers:

```
setComponentProp(Properties p, DataListener dl)
listenFor(CompFilter cf, MsgFilter mf, AddData ad, MsgListener ml, int l)
send(Message m)
```

where the `DataListener dl` and the `MsgListener ml` are pointers to callback functions invoked when additional data and messages arrive, respectively.

The `setComponentProp` operation allows sensors to specify the *properties* that describe the context in which they are, e.g., the fact that they are installed on a pig or a cattle, and the age of the animal.

The `listenFor` operation allows sinks to express their interests, by specifying both the content of the messages they are interested in (through the *message*

filter `mf`) and the sources they consider relevant (through the *component filter* `cf`). As an example, a sink could be interested in receiving messages such that: `activity!="stationary"` (the message filter), originating from sensors such that: `type==cattle AND age<24` (the component filter). The *additional data* `ad` is blindly transported by the protocol from the listening sinks to the matching sensors (those whose properties match the component filter). As an example, this data could be used by a sink to spread around information about the sampling period for sensing. Finally, the integer `l` is the *lease time* after which the expression of interest expires.

The `send` operation allows messages to be sent to the interested sinks, if any.

2.2 The Protocol in General

To support the mobile, multi-sink scenarios it has been conceived for, CCBR abandons the traditional approach to routing, which uses link-layer unicast packets to transport data from hop to hop, to use the broadcast facility provided by wireless networks. It also turns away from the usual, deterministic, sender-based approach to routing, to adopt a probabilistic mechanism to decide if and how packets are forwarded, leaving this decision to the receiver of the packet, which operates autonomously w.r.t. the sender.

All these choices, which differentiate CCBR from the previously proposed routing protocols for WSNs, were strongly influenced by our experience with mobile ad-hoc networks [4–7], which convinced us that "broadcast", "probabilistic", and "receiver-based" are the right keywords when mobility and multicast interactions (resulting from the presence of multiple sinks) enter the picture.

To describe CCBR in details we first describe its forwarding mechanism, i.e., how packets flow from sensors to sinks using the various routing tables, then we describe how such tables are built and maintained.

2.3 Forwarding

In CCBR, each sink has an associated *sink number*: an integer in the range $1...K$ where K is the maximum number of allowed sinks (see Sect. 2.4 for details on choosing sink numbers). To forward data from sources to sinks, each node maintains a *distance table* and a *content table*. The former stores an estimate of the distance (in hops) of the node from each sink, while the latter keeps track of the interests of sinks that are relevant for the node. In particular, the content table of a node with properties p maps each sink number n with those (not yet expired) message filters issued by n whose component filter matched p.

Figure 1 shows the content table of a sensor N with properties p when two sinks S_1 and S_2 (with sink numbers n_1 and n_2, respectively) have invoked the `listenFor` primitive, with message filters mf_{S1}, mf_{S2} and component filters cf_{S1}, cf_{S2}, both matching p. Note that N's content table does not include any information about sinks, like S_3 in the figure, whose interests do not match N's properties (i.e., cf_{S3} does not match p). This highlights the positive consequence of keeping context and content information separate: the routing tables can be

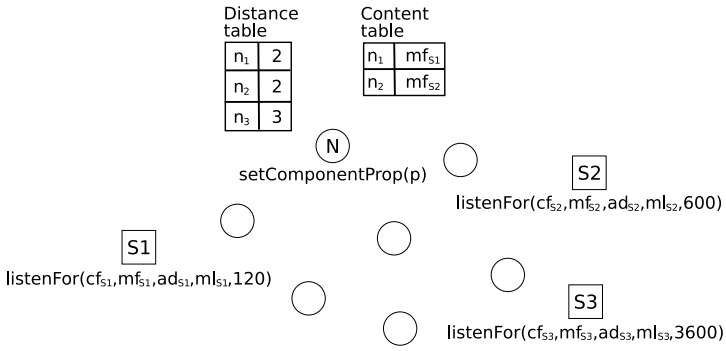

Fig. 1. Content and distance tables in CCBR.

kept smaller and they have to include the message filters but not the context filters. This saves memory and reduces the matching effort (and related power consumption) at message sending time, two fundamental issues in WSNs.

Whenever the `send(m)` primitive is invoked at a node N, the CCBR protocol looks up m's content in the content table and computes the set of sinks interested in receiving m. After that, it builds a *forwarding header* composed of a unique message identifier, a *destination vector*, and a *distance vector*.

- The destination vector is a bit-vector with length K, having a 1 in each position that corresponds to the number of an interested sink (as computed from the content table). In our example, supposing m matches mf_{S1} but not mf_{S2}, the destination vector has bit n_1 set, the others clear.
- The distance vector is an array of bytes, one for each interested sink, storing the distance of N from that sink (in order of sink numbers). In our example, it includes a single byte with an initial value of 2.

Afterward, CCBR builds a *forwarding packet* putting together the forwarding header and the message m and yields it to the MAC for broadcast transmission.

The first time a node receives a forwarding packet p, it compares the destination and distance vectors in the header of p with its own distance table. If the receiving node is farther or equally distant from the recipients of p, it drops it. If the receiving node is closer to at least one of the sinks listed in p's forwarding header: (i) it updates the distance vector in p, putting its own distance for those sinks it is closer, and (ii) it schedules the packet for transmission. The packet, indeed, is not transmitted immediately, while a *delay and cancel* mechanism is exploited[1].

Delay and Cancel. The packet is put in a *transmission queue*, where it remains for a period d_{tx} (the *delay of transmission*), which is smaller when the global improvement performed on the packet's distance vector is higher. As an example,

[1] To maximize the network lifetime we also add a probability of re-forwarding based on the remaining capacity of the node's battery.

consider the case of a node N with distance table $\{1|4,\ 2|2,\ 3|5\}$. When N receives a packet with destination vector 111 and distance vector $\{5,\ 3,\ 4\}$, it rewrites the latter putting $\{4,\ 2,\ 4\}$, and calculates a global improvement $H = 2$. At this point N will schedule the packet for retransmission with a delay $d_{tx} = \delta \cdot [\max(0, H_{max} - H) + rnd(0, 1)]$, where δ is proportional to the average time to transmit a packet (including MAC and transmission delays), while H_{max} has to be chosen looking at the average number of destinations for each packet. Whenever a node receives a packet, it looks at its transmission queue and deletes pending retransmission of the same packet (same identifier) if the queued copy has a distance vector which is higher or equal to the distance vector of the received packet for all destinations.

Under ideal conditions, this results in an efficient, greedy forwarding algorithm, which: (i) suppresses redundant transmissions (i.e., those originating from nodes equally distant from all the destinations); and (ii) favors, as forwarders, the nodes with the lowest distance from the highest number of destinations (i.e., the paths that lead to multiple destinations). In real scenarios, these results are only partially achievable since not all the receivers of a packet may hear each other, which may result in multiple path forwarding. We will come back on this issue while evaluating CCBR's overhead.

Mobility and Local Minima. Mobility is another factor that may break the mechanism above, by producing local minima in the distance function. This occurs whenever a node N has a wrong estimate of its distance from a sink, e.g., because it was once closer to it but now moved in a region where the real distance is higher. Nearby nodes will not forward packets generated by N because of the (wrong) lower distance it puts in the distance vector of those packets.

To solve this issue we complemented the basic forwarding algorithm above with a *retransmission mechanism*. After transmitting a packet (either as a source or as a forwarder), a node N puts it in a *retransmission queue*. If a predefined *timeout of retransmission* expires without hearing the same packet with at least one element in the distance vector lowered (i.e., if no one re-forwards the packet toward a destination), the node N:

1. adds a *retransmission bit-vector* to the forwarding header of the packet, with a one for each sink whose distance was set by itself;
2. increases the distance vector of the packet for each sink in the retransmission bit-vector;
3. transmits the resulting packet again.

Nodes hearing such a packet will reconsider it even if they already received it before, but only for improvement with respect to the sinks listed in the retransmission bit-vector (which is reset to 0, afterward).

The consequence of this mechanism is twofold: on one hand it increases the CCBR's resistance to collisions, which is good since link-layer broadcasting is particularly subject to collisions. On the other hand, increasing the distance vector for packets that where not forwarded by neighbors (step 2 above) also allows overcoming local minima, by increasing the set of potential forwarders for the retransmitted packet.

Unfortunately, asymmetric links may trigger this retransmission mechanism even when it was not required, thus increasing the network traffic without any positive effect on delivery. To limit this problem we allow each node to retransmit each packet at most once. Moreover, we also introduce in CCBR a mechanism of *credits*, which further reduces retransmission. When a packet is created, it is assigned a predefined credit: an integer stored into its forwarding header, which is decremented each time the retransmission timeout expires at a node. The retransmission mechanism does not apply to packets which ended their credit, i.e., the initial credit of a packet represents the maximum number of times the retransmission mechanism may fire along its route from the sender to the sink.

Delivery to Sinks. Whenever a sink receives a packet p, it looks at the bit in p's destination vector that corresponds to its sink number. If the bit is set the packet is forwarded to the application layer (by invoking the corresponding `MsgListener`) otherwise it is not. In any case, p is also processed normally for forwarding (sinks are standard nodes, so they must participate in the forwarding process). Finally, if p was targeted to that sink and it is not scheduled for forwarding, a special packet is created and broadcasted to stop the retransmission mechanism at the sender node.

2.4 Routing

To build and maintain distance tables, each sink periodically (every t_b seconds) broadcasts a beacon that contains its sink number, a sequence number and a distance, initially set to 0. Each node receiving a beacon, first increments the included distance and uses it to update its distance table, then it schedules the beacon for forwarding. Even in this case we use a "delay and cancel" mechanism to limit redundant transmissions. This time we are interested in favoring beacon retransmission by nodes that are farther away from the previous forwarder, to cover more distance with fewer retransmissions. Accordingly, we use a delay that is inversely proportional to the RSSI information provided by the MAC.

Whenever the `setComponentProp` primitive is invoked at a sensor, the CCBR layer simply stores the component property and the `DataListener` internally.

When the `listenFor` is invoked at a sink, its parameters (component filter, message filter, additional data, and lease time), are stored in a *filter table*. Every t_f ($t_f \geq t_b$) seconds the filters and additional data in such table for which the lease time is not elapsed are grouped and piggybacked on top of the next emitted beacon. When a sensor N receives this special "fat" beacon, it updates its distance table but also its content table as follow: for each filter whose component filter matches N's properties, the message filter, together with the sink number, are stored into the content table. If they were not already there (i.e., the same information has not been already received before), the additional data is passed to the related `DataListener`. Moreover, old filters from the same sink that are not refreshed by the beacon are deleted from the content table as this means that they expired.

Finally, to decide its sink number, at startup time each sink waits at least t_b seconds (but possibly a multiple of that) to see the beacons coming from other

sinks. Afterward, it randomly picks a number in the interval $1...K$ (see beginning of Sect. 2.3), not chosen by other sinks and starts operating. Clashes in choosing sink numbers may still happen (e.g., if two sinks turn on at the same time). They can be easily resolved by letting the sink with the lowest MAC address (or any other distinguishing value) to change its number when it discovers the clash.

3 Evaluation

Due to the difficulty of extensively testing a protocol like CCBR in a real setting, with hundreds of nodes moving around, and to do so in a replicable way, we decided to use a network simulator. Accordingly, we implemented the entire CCBR protocol (and a model of the CC2420 card and 802.15.4 MAC) in OMNeT++ [8], using the Mobility Framework [9] to simulate a mobile environment. We used a path loss channel model fully considering interferences from other, parallel transmissions to calculate (at run-time) the SNR of each frame.

Besides measuring the performance of CCBR under different conditions, we were also interested in comparing it with other protocols. In particular, we chose two simplified protocols, which well represent two very different classes of solutions to route packets in a mobile network. We call them *Gossip* and *Uni*.

Gossip is a structure-less protocol in which nodes send packets using the broadcast facility provided by the MAC layer and forwards them based on a pure probabilistic decision: when a node hears a packet for the first time it retransmit it with a probability $p \in (0, 1]$. Gossip is interesting as it represents the simplest possible approach to manage a mobile network, indeed it is often used as a baseline to compare protocols for MANETs.

Uni adopts a totally different approach. It uses beacons flooding the network as in CCBR to build unicast routing tables toward the sinks. Messages matching the interests of a sink S (considering both its context and content part) are sent, using link-layer unicast transmissions, hop by hop, up reaching S (different sinks are managed separately). Unicast is a good representative of those protocols, like Directed Diffusion [2], which set up a tree along which sources report their data to sinks.

To evaluate the performance of CCBR under different conditions, we considered a wide range of scenarios by changing the different parameters of our simulation: the density of the network (number of sensors per Km^2), the number of sinks (one of them stays firm at the center of the field, the others move around), the area of the field in which nodes move, the pattern of mobility (including the speed at which they move), the frequency at which each sensor sends out messages, and the selectivity of filters. In all the resulting scenarios we measured the performance of CCBR, Gossip, and Uni varying their key parameters: for CCBR the beaconing interval (the filters summarizing the interests of sinks are added to such beacons once every three of them) and the initial number of credits; for Gossip the forwarding probability; for Uni the beaconing interval.

This very extensive analysis resulted in a large body of data (and graphs), which allowed us to examine all the aspects of CCBR and how it performs w.r.t. Gossip and Uni. On the other hand, the limited space available does not allow

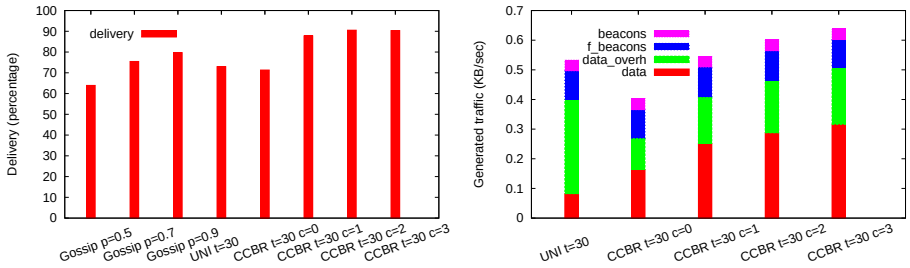

Fig. 2. The results measured in the default scenario.

reporting here all the tests we have done. Next sections discuss a subset of them, those we found most relevant[2].

3.1 The Default Scenario

Our default scenario reflects the situation of a set of persons or animals moving around in a limited area and being monitored by both mobile and fixed sinks. In particular, we consider an area of 0.5 Km^2, 50 sensors, and three sinks. Nodes (i.e., all the sensor nodes and two sinks over three) move according to a random waypoint mobility model with a speed between 1 and 2 m/s and a stop period of up to 10s. Sensors send a message every 10s, while the interests of each sink have 10% of chances to match each of the published messages, which means that 28% of the messages sent has to be delivered to at least one of the three sinks (i.e., Uni and CCBR only have to deliver one message every 36 seconds, on average).

Figure 2 (left) shows the delivery we measured for every protocol in this scenario. CCBR with zero credits, Uni, and Gossip with $p = 0.7$ provide similar results, correctly delivering between 72% and 74% of messages, while CCBR with the retransmission mechanism in place (i.e., when credits are greater than zero) provides the best performance, reaching a very good 90% of delivery with two credits. To put these numbers in context, we notice that the range of communication resulted, from our model, around 100m in absence of any other communication (i.e., no interferences), being much shorter when several nodes transmit data at the same time, as it happens in our scenario. As a result, our default density of 100 nodes per Km^2 results in periodic partitions of the network, which explains why none of the protocols we considered reaches a 100% of delivery.

As a further observation, it could appear strange how Gossip with a high probability of forwarding (e.g., $p = 0.9$) does not provide the best delivery. To understand why this happens we observe (graph not reported for space reasons) that Gossip generates 15 times more traffic than CCBR and Uni: 7.7 KB/s for

[2] The reader is warned that, even if we did not plot the confidence intervals in the graphs below (to avoid cluttering them), we took them into consideration. In particular, we run each simulation several times, varying the seeds of the random number generators we used in our models, until the size of the 95% confidence interval of the sample mean we measured was below 5% of the mean itself.

Gossip with $p = 0.9$ vs. 0.53 KB/s for Uni, thus incurring in a number of collisions and interferences that strongly limits its capacity of delivering messages.

A more detailed analysis of the traffic (measured at the physical layer) generated by CCBR and Uni in the default scenario is provided by Fig. 2 (right). First, we observe how the traffic generated by CCBR increases less than linearly with credits: a very positive result that proves the efficiency of the CCBR's retransmission mechanism. We also notice how beacons, including those carrying filters (i.e., `f_beacons` in figure), contribute around one fourth of the overall traffic. This is reasonable since in the default scenario each sensor produces, on average, one message every 36 seconds that is worth transmitting to at least one of the sinks, while the three sinks emit one beacon every 30 seconds which flood the network (albeit with the efficient "delay and cancel" mechanism explained above) to keep tables up to date, despite of mobility.

The last thing worth observing is how the total traffic generated by Uni is much greater than that generated by CCBR with zero credits, which performs comparably w.r.t. delivery. This is counterintuitive, since in the default scenario, in which most of the messages are addressed to a single sink, a routing based on unicast communication and shortest path tree forwarding, like Uni, should generate the least traffic to deliver messages. To understand why this happens, in Fig. 2 (right) we distinguished between the traffic generated by each protocol and sent to the MAC layer ("data" in figure), and the overhead generated by the MAC and physical layers ("data overhead" in figure). If we look at the first number alone, we see how Uni produces much less traffic than CCBR at the "routing" layer (above the MAC). Unfortunately, it incurs in a lot of overhead at the MAC and physical layers. This can be explained by remembering that 802.15.4 is a "reliable" MAC, using acknowledgements and multiple retransmissions to correctly deliver unicast packets. In presence of mobility this approach is detrimental, since the MAC wastes a lot of bandwidth in trying to reach nodes that went out of range. Conversely, CCBR uses broadcast at the MAC layer, which is unreliable but incurs much less overhead. This result supports our belief (see Sect. 2.2) that using broadcast and leaving the decision of forwarding to the receiver of a packet, not to the sender, whose routing tables may be outdated, is the right choice in presence of mobility.

3.2 The Impact of Credits and Beaconing Interval vs. Speed

CCBR was especially designed to support mobile scenarios, so it is important to see how it behaves under different patterns of mobility and how the beaconing interval and the number of credits impact its performance. In particular, while keeping all the other parameters as in the default scenario, we considered three levels of mobility: the default one, a "medium speed" scenario with nodes moving between 3 and 5 m/s and stopping for at most 2 seconds, and a "fast" scenario with nodes continuously moving at a speed between 5 and 10 m/s.

In Fig. 3 we plot the delivery and the *routing cost*[3], measured in KByte of traffic (generated at the physical layer) per delivered message, in these three

[3] Notice how the graphs plotting the routing cost are in logarithmic scale.

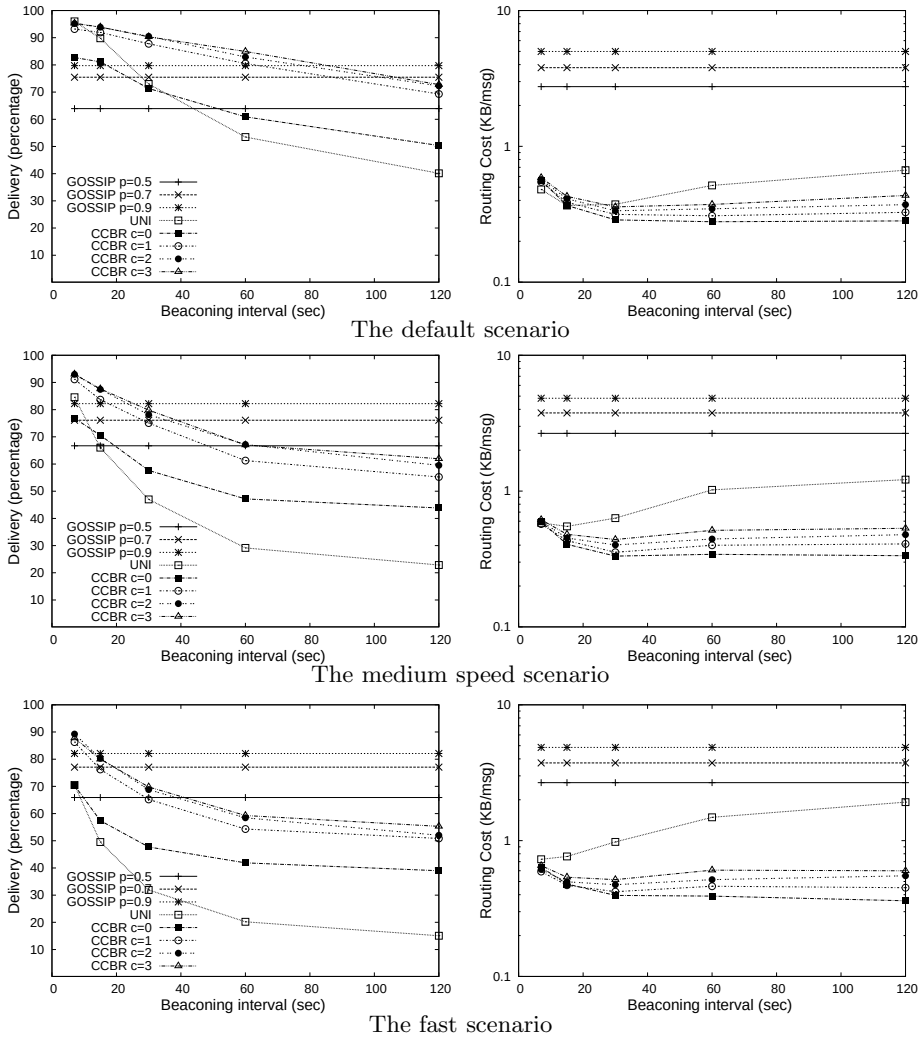

Fig. 3. The impact of speed and beaconing interval on results.

scenarios. First we may notice how, in all scenarios, Uni is the protocol which suffers more when the beaconing interval grows. On the contrary, the delivery of CCBR, even with zero credits, i.e., with the retransmission mechanism disabled, decreases much more slowly.

The efficacy of the retransmission mechanism can be measured by looking at CCBR with one or more credits. Its delivery is much better w.r.t. the case with zero credits, with the impact of the beaconing interval becoming less and less relevant as the number of credits grows. In the default scenario, CCBR with two credits provides 75% of delivery even with one beacon every 120 seconds, while in the hardest scenario of mobility one beacon every 30 seconds is enough

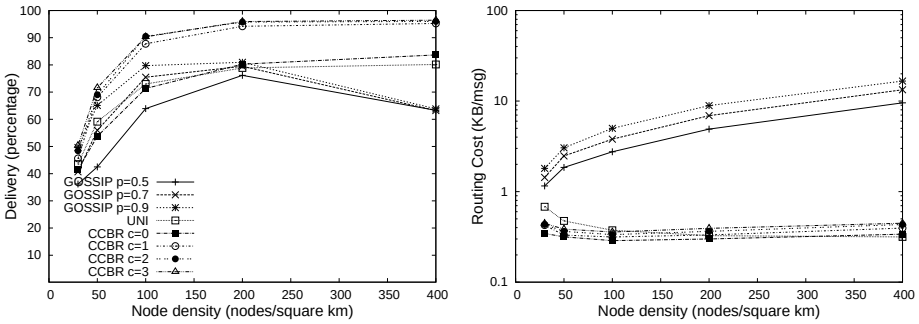

Fig. 4. The impact of node density on results.

for CCBR with two credits to provide a reasonable 70% of delivery. In the same situation the delivery of Uni is below 33%.

The graphs on the right in Fig. 3 show how CCBR, in every scenario of mobility we considered, is much better than Uni at keeping the routing cost constant while the beaconing interval grows. This, again, shows how a sender-based, unicast approach to routing, with automatic retries at the MAC layer, is not well suited to propagate packets in presence of mobility, i.e., when the correctness of the routing tables cannot be guaranteed. The same graphs also show how the number of credits has a minimal impact on the routing cost, a measure of the efficiency of the retransmission mechanism.

3.3 Density of Nodes and Area of the Network

Figure 4 shows the impact of a changing density of nodes when all the other parameters remain fixed at their default values. We observe that all protocols except Gossip increase their delivery as the density grows. Gossip performs better when density grows but only up to a certain limit (200 nodes per Km^2). After that point collisions become an issue even for the lowest forwarding probability we tested.

To compare the three protocols we may notice how CCBR with zero credits, Uni, and Gossip with $p = 0.7$ provide similar performance up to 200 nodes per Km^2, with CCBR and Uni also performing similarly up to the maximum density. On the other hand, the delivery tells only one side of the story. If we look at the routing cost we notice how Gossip with $p = 0.7$, while delivering the same percentage of messages of CCBR and Uni, uses much more bandwidth (the graph is in logarithmic scale). CCBR with zero credits and Uni incurs in very similar costs, with the former performing better when the density is low.

Even in this case, the retransmission mechanism proves its efficacy and efficiency. Indeed, CCBR with one or more credits outperforms all the other protocols at every density, while showing a routing cost that is only slightly greater than that of CCBR with zero credits and Uni.

Finally, we notice how the routing cost for CCBR is very marginally influenced by the density of the network. This is a very positive result for CCBR,

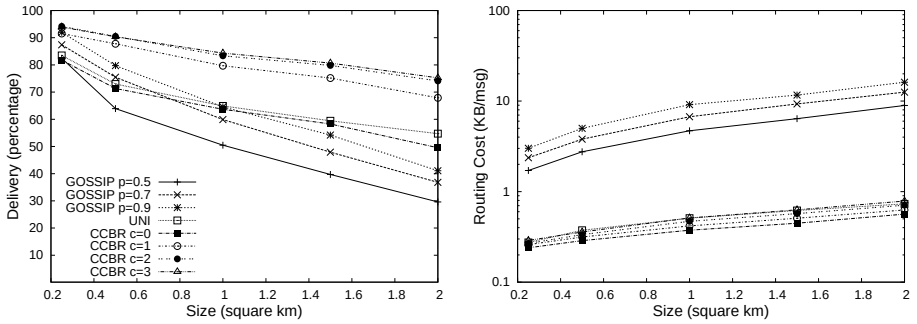

Fig. 5. The impact of an increasing area on results.

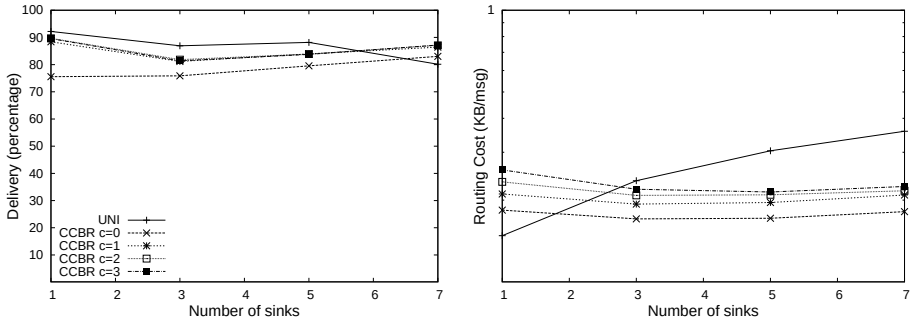

Fig. 6. The results for a static scenario (firm nodes).

that must be ascribed to the "delay and cancel" mechanism of forwarding, which minimizes useless retransmissions when the density grows.

Another interesting question to answer is how the performance of the different protocols changes when the area of the network grows (at a constant density of nodes). What we expect is an increase in the cost to deliver each packet, since the number of hops to travel increases, and this is confirmed by our tests (see Fig. 5), with CCBR and Uni decreasing their efficiency less than Gossip.

As for delivery, it decreases much more slowly when CCBR and Uni are adopted instead of Gossip, with CCBR outperforming Uni especially when the retransmission mechanism is used (i.e., with one or more credits). In a field of 2 Km^2 CCBR with two credits delivers 36% more messages than Uni, i.e., from 55% to 75% of delivery. The same figure also shows how the larger is the network the better it is to use more credits. This can be explained by observing that the more hops a packet has to travel the more credits it requires to overcome possible problems (i.e., local minima).

3.4 Static Scenario and Multiple Recipients per Message

After seeing how CCBR behaves in the mobile scenarios it was designed for, we are interested in seeing how it performs in a static scenario, and how it compares with Uni, which should operate at best when nodes are firm. Since CCBR was

developed to support multi-sink scenarios, we are interested in seeing how the number of message recipients impacts its performance in this stationary scenario.

Accordingly, we measured the performance of CCBR and Uni with firm nodes, letting each message go to every sink and progressively increasing the total number of sinks. Figure 6 shows the results we had. As we expected, Uni provides the best delivery in this scenario, with CCBR being very close and surpassing Uni when the number of sinks grows over a certain limit (at that point the contention on the channel to deliver the same packet to multiple sinks becomes an issue for Uni). What is even more positive is to observe how the cost of routing for CCBR is initially greater than that of Uni, but becomes smaller as soon as the number of receivers of each message grows. This shows that we centered our second goal: CCBR is capable of optimizing routing when the same message has to be delivered to more sinks.

As a final remark, by comparing the behavior of CCBR and Uni in the single sink, static scenario (the hardest one for CCBR), we may evaluate the efficiency of the "delay and cancel" mechanism. Number at hands, we measured (for CCBR with zero credits) 24% more traffic than Uni. This means that once every four hops the delay and cancel mechanism "fails". A good result considering that this mechanism has been designed to provide the robustness and efficiency required for very different scenarios, i.e., those including mobility and multiple sinks.

4 Related Work

While fully mobile WSNs have been rarely considered so far, the case of mobile sinks in a stationary WSN is not new [10–12]. Sink mobility has been also considered as a way to transform the problem of routing data toward the sinks into the problem of routing sinks towards the data [13–16].

The case of WSNs involving mobile sensors, like those deployed to monitor animals [17, 18], is also not new. Unfortunately, the routing protocols proposed in this area [19–22] focused on strongly disconnected scenarios, where the very low density of nodes requires mechanisms typical of delay-tolerant networks. As mentioned, we consider different scenarios in which sink reachability is more the rule than the exception, thus allowing real-time monitoring of critical situations.

These scenarios are typical of Mobile Ad-hoc NETworks (MANETs), indeed CCBR is based on our previous experience in developing content-based publish-subscribe routing protocols for MANETs [4–7]. However a direct application of these and other content-based routing proposals for MANETs (e.g., [23]) to WSNs is hard because of sensors' tight resource constraints.

To the best of our knowledge CCBR is the first data-aware (i.e., content-based) routing protocol especially designed for mobile WSNs. Other protocols adopting a similar, data-aware model, such as Directed Diffusion (DD) [2], GRAB [24], and TinyCOPS [3] have considered fixed networks as their reference scenario [25]. An evaluation of these protocols in mobile scenarios will be beyond their intended scope, even unfair [26], since they only consider (i) long-term faults with a frequency in the range of hours or days, due to slow fading and node failures, (ii) short-term faults with frequency in the order of μs due to

transmission errors and collisions. As an example, DD handles the former with path repairing mechanisms, while delegates the latter to unicast, reliable MACs. Mobility dynamics, being in the order of seconds, sits almost in the middle of these two extremes, and cause faults that are too frequent to be handled by repair mechanisms and too long lasting to be handled by retransmission. Moreover, as also noted in [27] and confirmed by our simulations, unicast and reliable MACs aggravate the situation, by interpreting faults due to mobility as transmission errors, causing useless delays and wasting bandwidth and energy.

GRAB strives for higher robustness with respect to both short and long-term faults by abandoning sender-based, unicast forwarding in favor of link layer broadcast primitives and receiver-based forwarding. Differently from CCBR, however, to increase robustness it does not suppress redundant retransmissions: each node within a certain, progressively reducing, distance from the sink (resulting in a lens-shaped area) retransmits the message. Unfortunately, the distance field used by GRAB for taking receiver-based forwarding decisions rapidly deteriorates when nodes move. As before, while GRAB approach is to rebuild the cost field in those situations, it lacks mechanisms to mask faults induced by mobility, such as the CCBR retransmission and credits mechanism.

Other mechanisms adopted by CCBR are receiver contention, channel overhearing and RSSI estimations. They exploit WSNs peculiarities to allow distributed routing decisions with minimal state and communication. Other proposals in WSNs use similar mechanisms. Receiver contention is used by geographic routing [28, 29, 26], opportunistic routing [30], data dissemination [31], and cooperative diversity [32] schemes to select the best forwarder or relay with minimal overhead. CCBR uses this approach with a different goal: to select efficient routes leading to multiple destinations in spite of unreliable routing information resulting from mobility. Channel overhearing is used in geographic routing to trigger void avoidance phases, while CCBR (using hop-counts) does not have voids but uses instead overhearing to trigger its local minima escape mechanism. Finally, exploiting RSSI to quickly establish distances was already proposed in [33].

5 Conclusion

While the typical application for WSNs is in environmental monitoring, different scenarios are also possible. In particular, in the WASP project financed by the EU Commission we are considering scenarios like controlling animals in a farm or monitoring elder people in hospices, which involve mobile nodes and multiple sinks, while continuous connectivity is guaranteed by the small area in which sensors move (e.g., the field in which cattle move) or by the presence of fixed sensors that may act as forwarders (e.g., in a house).

The CCBR protocol we described in this paper provides a context and content-based routing layer especially tailored to such scenarios. On top of this layer, it becomes easy to develop several communication paradigms, from publish-subscribe to continuous queries ala TinyDB. Our simulations show that the mechanisms used by CCBR are very effective in providing good delivery with a low cost of routing, which potentially implies a low power consumption.

With respect to the issue of power consumption, other partners of the WASP project developed a MAC protocol optimized for the kind of broadcast communication adopted by CCBR. It guarantees low power drain both for sending and for receiving broadcast packets by extensively using advanced duty cycling mechanisms. We are currently implementing CCBR on top of the first release of this MAC to measure power consumption on real nodes.

As a further work in this area, we are interested in investigating how to add "in network processing" capabilities to CCBR, to allow sinks to specify some aggregation function, letting CCBR decide where to aggregate data. A very complex task in mobile scenarios like those we target.

References

1. Carzaniga, A., Wolf, A.L.: Content-based networking: A new communication infrastructure. In: Proc. of the NSF Workshop on an Infrastructure for Mobile and Wireless Systems. Number 2538 in LNCS (2001) 59–68
2. Intanagonwiwat, C., Govindan, R., Estrin, D., Heideman, J., Silva, F.: Directed diffusion for wireless sensor networking. Trans. on Netw. **11**(1) (Feb 2003) 2–16
3. Hauer, J.H., Handziski, V., Kpke, A., Willig, A., Wolisz, A.: A component framework for content-based publish/subscribe in sensor networks. Wireless Sensor Networks (2008) 369–385
4. Mottola, L., Cugola, G., Picco, G.P.: A self-repairing tree topology enabling content-based routing in mobile ad hoc networks. IEEE Trans. on Mobile Computing **7**(8) (2008) 946–960
5. Baldoni, R., Beraldi, R., Querzoni, L., Cugola, G., Migliavacca, M.: Content-based routing in highly dynamic mobile ad hoc networks. Int. Journ. of Perv. Comp. and Comm. **1**(4) (2005) 277–288
6. Costa, P., Migliavacca, M., Picco, G.P., Cugola, G.: Epidemic algorithms for reliable content-based publish-subscribe: An evaluation. In: ICDCS. (2004) 552–561
7. Picco, G.P., Cugola, G., Murphy, A.L.: Efficient content-based event dispatching in the presence of topological reconfiguration. In: ICDCS. (2003) 234–243
8. : OMNeT++ Web page `www.omnetpp.org`.
9. : Mobility Framework for OMNeT++ Web page `mobility-fw.sourceforge.net`.
10. Luo, H., Ye, F., Cheng, J., Lu, S., Zhang, L.: Ttdd: Two-tier data dissemination in large-scale wireless sensor networks. Wireless Networks **11**(1-2) (2005) 161–175
11. Kim, H.S., Abdelzaher, T.F., Kwon, W.H.: Minimum-energy asynchronous dissemination to mobile sinks in wireless sensor networks. In: SenSys. (2003)
12. il Hwang, K., In, J., Eom, D.S.: Distributed dynamic shared tree for minimum energy data aggregation of multiple mobile sinks in wireless sensor networks. In: EWSN. (2006)
13. Shah, R., Roy, S., Jain, S., Brunette, W.: Data mules: Modeling a three-tier architecture for sparse sensor networks. In: IEEE SNPA Workshop. (2003)
14. Somasundara, A., Kansal, A., Jea, D., Estrin, D., Srivastava, M.: Controllably mobile infrastructure for low energy embedded networks. Mob. Comp., IEEE Transactions on **5**(8) (Aug. 2006) 958–973
15. Chatzigiannakis, I., Kinalis, A., Nikoletseas, S.: Efficient data propagation strategies in wireless sensor networks using a single mobile sink. Comput. Commun. **31**(5) (2008)

16. Ammari, H.M., Das, S.K.: Promoting heterogeneity, mobility, and energy-aware voronoi diagram in wireless sensor networks. IEEE TPDS **19**(7) (2008) 995–1008
17. Bonnet, P., Leopold, M., Madsen, K.: Hogthrob: towards a sensor network infrastructure for sow monitoring. In: DATE. (2006)
18. Butler, Z., Corke, P., Peterson, R., Rus, D.: Dynamic virtual fences for controlling cows. In: Experim. Robotics IX, Springer Tracts in Adv. Rob., Springer (2006)
19. Juang, P., Oki, H., Wang, Y., Martonosi, M., Peh, L., Rubenstein, D.: Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In: ASPLOS, San Jose, CA. (October 2002)
20. Pasztor, B., Musolesi, M., Mascolo, C.: Opportunistic mobile sensor data collection with scar. In: MASS. (2007)
21. Henriksson, D., Abdelzaher, T., Ganti, R.: A caching-based approach to routing in delay-tolerant networks. ICCCN 2007 (Aug. 2007) 69–74
22. Luo, L., Huang, C., Abdelzaher, T., Stankovic, J.: Envirostore: A cooperative storage system for disconnected operation in sensor networks. (2007)
23. Petrovic, M., Muthusamy, V., Jacobsen, H.A.: Content-based routing in mobile ad hoc networks. In: MobiQuitous. (2005) 45–55
24. Ye, F., Zhong, G., Lu, S., Zhang, L.: Gradient broadcast: a robust data delivery protocol for large scale sensor networks. Wirel. Netw. **11**(3) (2005) 285–298
25. Bokareva, T., Bulusu, N., Jha, S.: A performance comparison of data dissemination protocols for wireless sensor networks. GlobeCom Workshops 2004. IEEE (Nov.-3 Dec. 2004) 85–89
26. He, T., Blum, B.M., Cao, Q., Stankovic, J.A., Son, S.H., Abdelzaher, T.F.: Robust and timely communication over highly dynamic sensor networks. Real-Time Syst. **37**(3) (2007) 261–289
27. Heissenbüttel, M., Braun, T., Wälchli, M., Bernoulli, T.: Evaluating the limitations of and alternatives in beaconing. Ad Hoc Netw. **5**(5) (2007) 558–578
28. Zorzi, M., Rao, R.R.: Geographic random forwarding (geraf) for ad hoc and sensor networks: Multihop performance. IEEE Trans. Mob. Comput. **2**(4) (2003) 337–348
29. Heissenbüttel, M., Braun, T., Bernoulli, T., Wälchli, M.: Blr: Beacon-less routing algorithm for mobile ad hoc networks. Comp. Comm. **27**(11) (2004) 1076–1086
30. Biswas, S., Morris, R.: Exor: opportunistic multi-hop routing for wireless networks. In: SIGCOMM. (2005) 133–144
31. Mastrogiovanni, M., Petrioli, C., Rossi, M., Vitaletti, A., Zorzi, M.: Integrated data delivery and interest dissemination techniques for wireless sensor networks. GLOBECOM (2006)
32. Bletsas, A., Khisti, A., Reed, D.P., Lippman, A.: A simple cooperative diversity method based on network path selection. IEEE JSAC **24**(3) (2006) 659–672
33. Dutta, P., Culler, D., Shenker, S.: Procrastination might lead to a longer and more useful life. In: 6th Workshop on Hot Topics in Networks (HotNets VI). (2007)