Demo Abstract: A MAC Contest between LPL (the Champion) and Reins-MAC (the Challenger, an Anarchic TDMA Scheduler Providing QoS)

Matteo Ceriotti
Fondazione Bruno Kessler—IRST
Trento, Italy
ceriotti@fbk.eu

Amy L. Murphy
Fondazione Bruno Kessler—IRST
Trento, Italy
murphy@fbk.eu

LPL [5], or BoX-MAC in its TinyOS implementation, is arguably the most common MAC protocol for WSNs. Its extensive use in real world deployments is justified by a simple implementation, available online, that meets the requirements of a vast majority of monitoring applications. Despite the plethora of competitors, it still remains the reference in the MAC scheduling world. However, its random nature inherently hampers its possibility to support effectively any application driven Quality of Service requirement.

To provide these guarantees, we offer REINS-MAC, a TDMA-based MAC. While common TDMA solutions require each node to rigidly follow an agreed upon communication schedule, in REINS-MAC each individual defines its own slot inside the overall frame. Despite the common belief that such dynamic communication scheduling is infeasible [3], REINS-MAC both adapts to the current network topology, and allows each node to change the size and position of its slot. REINS-MAC achieves the aforementioned flexibility and anarchy by removing one fundamental parameter of TDMA: the network-wide slot size constant.

This demo establishes a contest between REINS-MAC and BoX-MAC under a variety of network conditions and different parameter settings of a multi-hop monitoring application. The audience is challenged to tune the BoX-MAC parameters such as the sleep interval and modify the network topology to create conditions that favor the CSMA solution. For each setup, the behavior of the two protocols will be scored on throughput, overhead, etc.

1 REINS-MAC

REINS-MAC builds on the literature of Pulse Coupled Oscillators (PCOs), which is inspired by firefly behavior [4]. Each beetle periodically emits a light. Other fireflies observe these flashes and alter the timing of their own flashes, eventually yielding a unison blinking of the whole swarm. Formally, the periodic behavior can be described as a cyclic *oscillator* manifest as a *phase* variable, θ, that *(i)* assumes a value in a given range, e.g., $[0,1)$, *(ii)* increases linearly over time and *(iii)* resets to the first limit when the second is reached. To mimic the flash of a firefly, resetting the phase variable can be combined with a *pulse* that is essentially a notification of the reset.

In fireflies, periodic pulsing and observing others is not sufficient to produce synchrony. This coupling is achieved by either actively anticipating or delaying the individual pulse. Mathematically, this means that the phase variable of a firefly is no longer a constant, linear function over time, but it exhibits spontaneous changes in response to observations. The rules driving these changes are captured in a *coupling function* that establishes the phase variable at which the flash will occur in the next oscillation round.

A distributed problem that can be reduced to the PCO scheme is *pulse scattering* [2], whose goal is to obtain pulsing schedules where all the nodes that can hear each others beating are evenly spread throughout the oscillation period. A simple but effective solution is to distance the local pulse as much as possible from the surrounding ones; this is achieved by placing the beating almost exactly in the middle between two adjacent perceived pulses.

1.1 TDMA Scheduling

Slot Allocation. We approach slot size selection and slot allocation by noting the similarity between this communication problem and that of pulse scattering. Consider the simple case when all nodes are within communication range. Nodes spread their pulses evenly throughout the frame, and if a transmission slot for a specific node is defined to begin at the moment at which it pulses and end when the next node pulses, a TDMA schedule emerges. Consider directly applying such an algorithm in a multi-hop network. While node *C* in Figure 1(a) will scatter with respect to both *A* and *D*, because *A* and *D* are not within range, they do not scatter with respect to one another and their slots may overlap. Simultaneous transmissions from *A* and *D* will overlap, causing collisions at *C*, as shown in Figure 1(b). This issue is solved by applying pulse scattering with a horizon of 2 hops as sketched in [1] and shown in Figure 1(c). When the node's own pulse fires, it can transmit; when a 1 hop distant pulse is received, the node listens for incoming transmissions; when a 2 hops distant pulse is expected the node sleeps to save energy, as no communication involving it is possible.

Politely Join the Schedule. We first recall that no synchronization is required in the solution presented above. In fact, a node wanting to join the system can trivially learn the current communication schedule by listening to the pulsing of its neighbors, and independently decide the best placement

SenSys'10, November 3–5, 2010, Zurich, Switzerland.
ACM 978-1-4503-0344-6/10/11

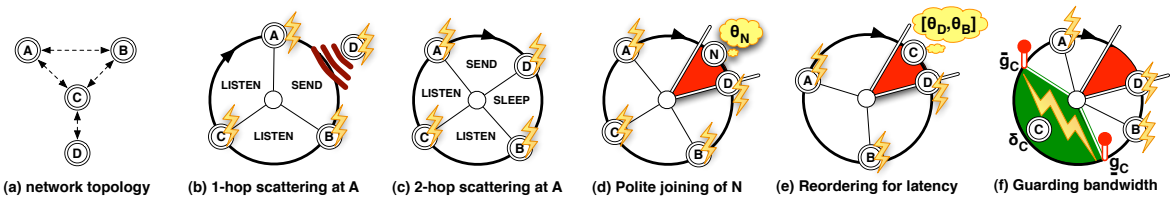

Figure 1. Core REINS-MAC functionality.

of its own pulse. However, this solution does not avoid collisions with ongoing communication. Therefore, we adopt the approach employed in classic TDMA solutions, allocating a coordination slot, common to all schedules. Augmenting our MAC-solution with such a synchronized slot is trivial: we simply require nodes to pulse twice. One pulse is controlled by the scattering equation as defined previously, while the other is updated according to a synchronization coupling function, e.g., align the local pulse to the average of the heard pulses. Notification of both pulses is combined and transmitted at the beginning of the assigned transmission slot. The synchronized pulse identifies a small, fixed size slot, shown as a shaded region in Figure 1(d), during which all nodes must listen, and any node can transmit using a simple contention-based access approach. For a node to join the schedule, it transmits in the control slot a deferred pulse indicating its *preferred* location in the schedule. All nodes incorporate the pulses indicated in the control messages into their schedules, and after a stabilization period, the new node can begin to transmit without disrupting ongoing communications.

1.2 QoS Provisioning

Latency Control. In the scheduling solution described thus far, the sequence of node pulses is fixed. This sequence affects latency, therefore enabling changes in the sequence of node pulses is the key to support latency constraints. Our solution is simply for a node to remove itself from the schedule, then re-enter the schedule at its desired location, using the control slot described previously. This is shown in Figure 1(e) as node C removes itself from its original location between A and B, and uses the control slot to announce its new, desired position between nodes D and B.

Bandwidth Reservation. The bandwidth available to each node is constrained to its own slot size. Therefore, REINS-MAC allows each node to increase the single, assigned slot to the size requested at run-time, δ. To accomplish this, we give the pulse a *duration* in time, delimiting the beginning and end of the pulse with *guarding* phases, g and $\overline{g}$, spread equal distance from the center of the pulse. This guarding phases act as actual pulses for the other nodes in the schedule, which scatter accordingly, as described for slot allocation. As depicted in Figure 1(f), by extending the pulse duration of C to begin at the first guard, g and end at the second, $\overline{g}$, the slots of A and B shift, leaving a slot of at least size δ_C that provides the requested bandwidth guarantee to C. In the example, the total reserved duration must be less than the distance between A and B, as placing the guards beyond these limits results in incorrect pulse scattering.

2 Demonstration

The demonstration will be composed of two sets of a dozen nodes each, one running LPL and the other REINS-MAC, on different radio channels. On top of both MAC protocols, a generic multi-hop routing protocol will gather protocol metrics from the entire network and fake application data from a predefined subset of nodes. The data will be collected at a PC, where a graphical interface will show the performance of each network. From the same interface, it will be possible to change both the LPL sleep interval and the application data rate. The audience will be given the possibility to physically relocate the nodes and change the LPL sleep interval. While running the experiments, several metrics will be computed, e.g. throughput, overhead, energy consumption, which will contribute to an overall score. The audience will compete to both produce the highest score for LPL, and find the most challenging setting for REINS-MAC.

3 References

[1] J. Degesys and R. Nagpal. Towards desynchronization of multi-hop topologies. In *Proc. of the 2nd Int. Conf. on Self-Adaptive and Self-Organizing Systems (SASO)*, 2008.

[2] A. Giusti, A. L. Murphy, and G. P. Picco. Decentralized scattering of wake-up times in wireless sensor networks. In *Proc. of the 4th European Conf. on Wireless Sensor Networks (EWSN)*, 2007.

[3] K. Langendoen. Medium access control in wireless sensor networks. In H. Wu and Y. Pan, editors, *Medium Access Control in Wireless Networks*, pages 535–560. Nova Science Publishers, Inc., May 2008.

[4] R. E. Mirollo and S. H. Strogatz. Synchronization of pulse-coupled biological oscillators. *SIAM J. Appl. Math.*, 50(6):1645–1662, 1990.

[5] J. Polastre, J. Hill, and D. Culler. Versatile low power media access for wireless sensor networks. In *Proc. of the 2nd Int. Conf. on Embedded Networked Sensor Systems (SENSYS)*, 2004.