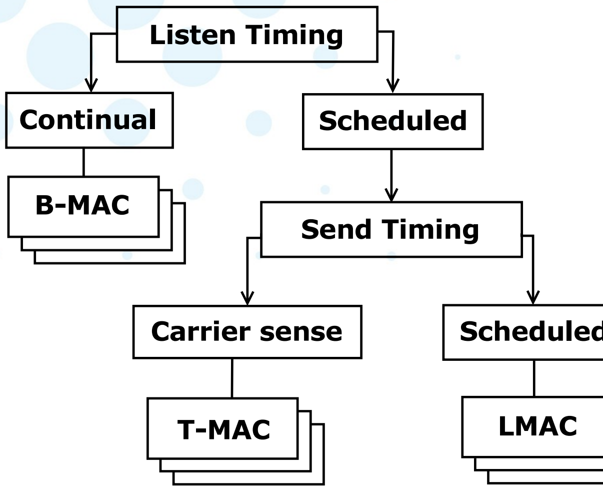
Towards component reuse in MAC protocols

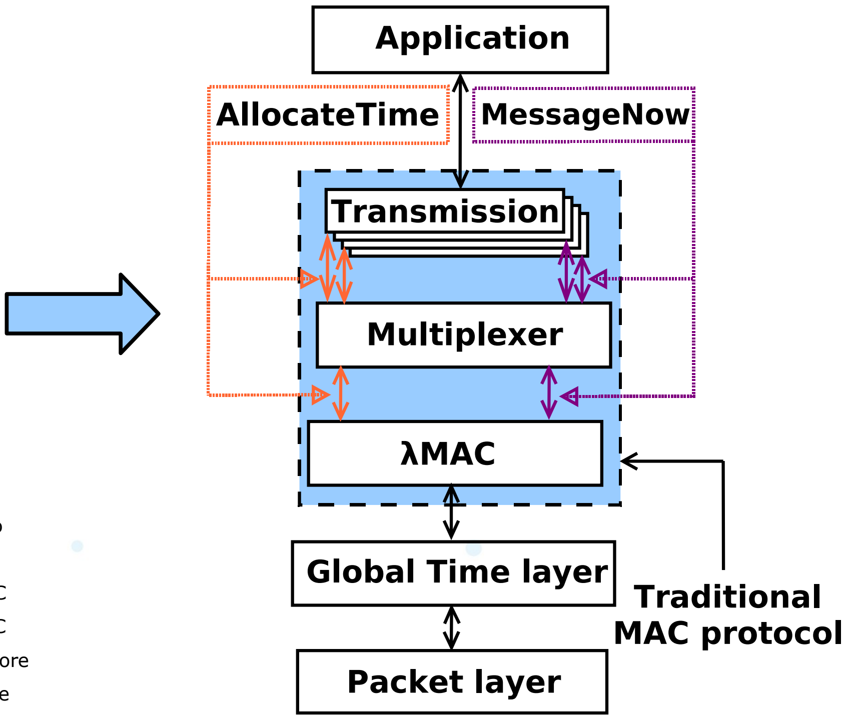
Tom Parker, Maarten Bezemer, Koen Langendoen

Current MAC Protocols



Most current WSN MAC protocol implementations have multiple tasks to perform - deciding on correct timing, sending of packets, sending of acknowledgements, etc. However, as much of this is common to all MAC protocols, there is duplication of functionality, which leads to larger MAC protocol code size and therefore increasing numbers of bugs. We therefore wished to redesign the process for creating a MAC protocol such that the common functionality that does not necessarily need to be in a MAC protocol itself can be separated out.

New MAC Stack

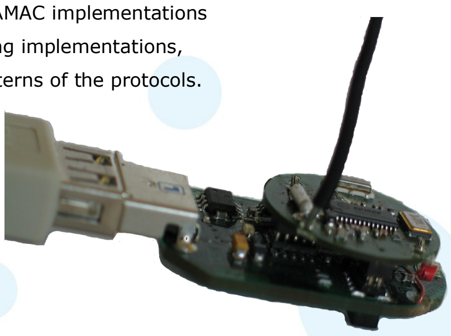


We redefined the required modules and connections as shown above.

Implementation

We implemented T-MAC, LMAC and Crankshaft, as well as a "trivial" test MAC in TinyOS. These new implementations have been tested both in TOSSIM and with our 26 node hardware testbed, as well as against existing implementations of the protocols (simulations for LMAC and Crankshaft; an earlier TinyOS version for T-MAC). Results from these tests were promising - the λMAC implementations behaved in the same way as the existing implementations, reproducing faithfully the expected patterns of the protocols.

The level of effort required for implementation was also much less; λT-MAC was only 32% of the size of the original T-MAC implementation, and Crankshaft was implemented from scratch in only a month.

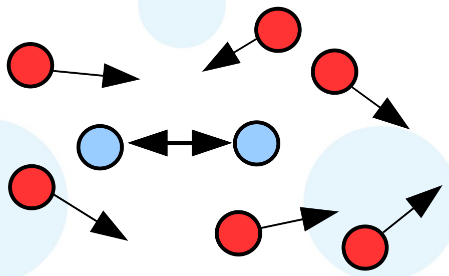


Program sizes

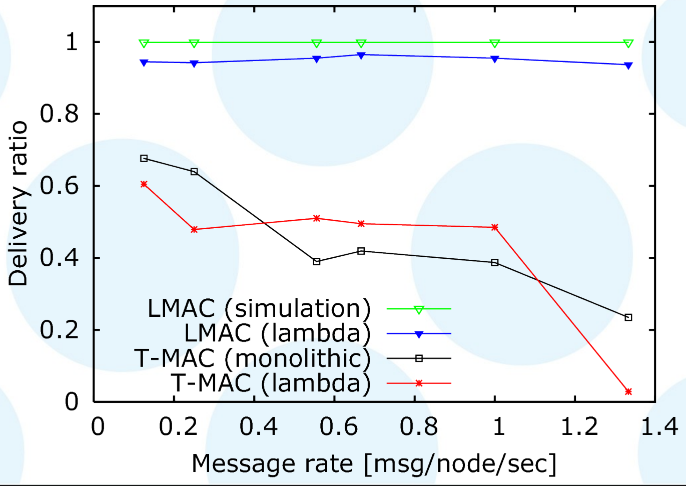
Component	Lines of Code	% of MAC Stack
MAC Framework	3961	n/a
λT-MAC	1426	26%
λLMAC	814	17%
Crankshaft	578	12%
Trivial MAC	339	7.8%

Cloud Experiment

We performed a series of benchmark tests on the new MAC layers. One of them used a cloud of broadcast nodes surrounding a unicast link.



The current performance of the unicast link for different MACs with the new framework and existing work is shown in the graph below.



TU Delft
Technische Universiteit Delft