Appeared in Proc. 1. GI/ITG Fachgespräch ``Sensornetze'' (Technical Report TKN-03-012 of the Telecommunications Networks Group, Technische Universität Berlin), pp. 10-17, Berlin, July 2003

A common wireless sensor network architecture?

Vlado Handziski, Andreas Köpke, Holger Karl, Adam Wolisz
Telecommunication Networks Group
Technische Universität Berlin
handzisk,koepke,karl,wolisz@ft.ee.tu-berlin.de

1 Fragmentation of research

Wireless sensor network (WSN) research is currently heavily fragmented: Independent developments take place on both sides of the Atlantic, in many different research groups. There is not yet a common *"lingua franca"* as in some other fields of networking research, where e.g. Linux on Intel processors or the Network Simulator [1] have achieved such a status—with all the advantages and disadvantages of such a predominance. In WSNs, the need for a harmonization of research efforts is particularly felt in the following areas.

Simulation environments Many different research groups use different simulation environments for performance evaluation. Currently popular tools include NS/2, Opnet, GlomoSim, Qualnet, or OmNET++. None of these simulation tools perfectly meet the demands of WSNs. For example, the wireless channel model is too simplistic and not easily changed (NS/2), not all relevant protocols are easily available (Omnet), some are commercial (Opnet, Qualnet), some are too infrequently used to easily allow comparison of results with work of other researchers (all tools with the possible exception of NS/2).

More importantly, there is no simulation tool for which convincing models of applications or sensor excitation are available.

Hardware platforms The overall setup of sensor node prototypes that have been developed and are currently used by different groups for experimental research is in principle very similar. Examples for such nodes include the Mica Motes, the nodes used by the EYES project [2], and nodes developed locally at Technische Universitt Karlsruhe, ETH Zrich, Freie Universitt Berlin, or Technische Universitt Dresden. Typical microcontrollers are the Atmel, Motorola microcontrollers or the Texas Instruments MSP 430; usable radio modems include those by RFM, Chipcon, Maxim, Nordic, RFMD, Xemics, Infineon and others; also, Bluetooth chipsets are sometimes used.

While all of these nodes have their individual advantages and disadvantages and the level of diversity is natural for the current state of WSN research, a lot of synergy could be gained if a convergence to a single prototype could be achieved. The economies of scales alone would make such an effort worthwhile by making prototype nodes available in larger quantities at low cost. The Mica Motes offerings by Crossbow are a step in this direction, but are still somewhat expensive. Possibly, the current developments in the IEEE 802.15.4 standardization process will create sufficient pressure towards unification.

Moreover, this multitude of prototypes often renders experimental results difficult to compare. Radio modems have different modes of operations with different power consumptions, resulting in differently optimized lower-layer protocols. While these solutions are important and necessary, what is still missing is the next logical step in comparing the design decisions for example for MAC layer protocols that result from various properties of a specific node.

Operating systems Different prototypes are beneficial as long as their idiosyncracies can be abstracted away if necessary. Usually, an operating system performs this role of hardware abstraction. The design decisions taken for various prototypes vary quite a bit: from a simple hardware abstraction over an run-time environment like TinyOS to a full-blown task scheduler like in the EYES OS.

Again, the problem of comparability arises: How can own solutions be compared with established solutions and protocols that have been developed for a given operating system?

Development environments Sometimes, the hardware abstraction/operating system mandates the use of given development environments like MS Windows or commercial compilers like IAR. Typically, publicly available tool chains are preferable.

2 Three approaches towards better integration

One important step for overcoming these overcomings would be a harmonized simulation and experimental environment, which is a somewhat utopian goal. Nonetheless, the following three issues could be attempted.

Hardware abstraction for new sensor node prototypes While a complete porting of an arbitrary operating system to a new hardware is a painful undertaking, it might be reasonable to look for a useful hardware abstraction set that could be supported by most or all of the existing and imagined sensor nodes, that would be expandable to integrate future possibilities, and on top of which operating systems or runtime systems could be easily ported or developed.

The TinyOS abstraction layer is an excellent start in this direction; also, some work done at Freie Universitt Berlin could be leveraged here.

The advantage for hardware developer is the wider range of possible environments which could use a new hardware; the advantage to software developers is the wider range of usable hardware.

Ideally, for each sensor node prototype in practical use, such a hardware abstraction layer should be made publicly available.

Abstract model of power consumption As different, heterogenous hardware is here to stay, still a fairly accurate and *standardized* model of the capabilities of hardware is required, with respect to mainly power/energy consumption and performance. Both the radio front end and the microcontroller of a sensor node should be included in this model. If the actual sensor or actuator hardware also noticeably contributes to power consumption (as is the case e.g. for magnetometer sensors), it should also be mentioned.

The most relevant aspects of such an energy/power consumption model are:

- How many modes of operation can be distinguished for the microcontroller (sleep modes, operation, potentially reduced clock rate, ...) and the radio front end (sleep, idle, receive, transmit, ...)? Are there any other functional parts for which relevant state transitions occur, and what are they?
- What is the power consumption for these blocks in each of these modes? Does it depend (in which way) on additional parameters, e.g., the modulation?
- In case transmission power control is available: What is the mapping from radiated power to consumed power? Does it depend on additional parameters, e.g., modulation?
- What is the transition time from each operational mode to each other one (if the transition is directly possible at all), what is the power consumed during such a state transition (and, hence, the energy)?

- What are the performance parameters of the radio modem, especially, the receiver sensitivity, the maximum output power, the blocking characteristics, the mapping from SINR to bit error rate (depending on data rate and modulation), the available data rate(s), number of available channels?
- What are additional quality parameters like temperature drift of the radio front end or frequency stability, calibration of received signal strength indication (RSSI) signals, etc.?
- Any additional "unusual" characteristics of the microcontroller or the radio modem, e.g., the possibility to emit a busy tone while receiving data? Is the level of forward error correction or channel coding controllable, etc.?

While most of this information is available in some form or another in manufacturer's data sheets, a single, easy to access repository of this type of information is missing. Such a repository should be quite valuable, especially when evaluating communication protocols by simulating, providing a reference to actually realistic assumptions about the behavior of a range of real hardware. Moreover, experience reports like "often bad frequency stability" would add to the usefulness.

A Protocol Architecture Scheme for WSN One reason that makes the implementation of wireless networks in general and wireless sensor networks in particular so challenging is the need to exchange information between functionalities that belonged, in the traditional ISO/OSI model, to different protocol layers. A simple example is link-layer triggering for handover in cellular networks, transmission-power-aware routing protocols in ad hoc networks, and the multitude of mutual influences of protocol functionalities in wireless sensor networks.

In such a situation, the reuse of communication protocols, especially from a thrid party, is challenging. Ideally, individual "building blocks" for wireless sensor network protocols should be identified and implemented separately, with different ideas for the mechanisms realizing the desired functionalities. These building blocks would pass packets between each other, but also exchange meta-information, e.g., concerning the topology of the network or about the geographic position of individual nodes. In this sense, there will be building blocks that are predominantly engaged in packet exchange (e.g., a link layer building block) while others will be mostly tasked with the collection and computation of meta information (e.g., a building block that provides estimates about a sensor node's position).

When trying to structure a WSN protocol suite along these lines, identifying the most relevant building blocks and their most important packet passing and information exchange relationships, a structure like the one in Figure 1 is a possible outcome (depicting an intermediate state of the EYES project's architecture discussion).

This approach results in two challenges: How to organize the passing of a packet to the correct next building block; the second challenge is how to then organize the information exchange between building blocks. The first challenge is akin to the same problem in conventional protocol stacks with well-known solutions like packetfilters [3] which should also be applicable here. The second one is more difficult since it is neither clear *when* information is required from another building block nor *to whom* to deliver information. As an example, a building block for location estimates might be used by additional building blocks even after this block has been completed.

Hence, a separation of communication in time and space is required—the publish/subscribe model is doing exactly this. Providers of information publish it under a given "name", users of information can subscribe to such names and be informed of any value changes. One example for such a name would be RSSI: the physical layer could publish this information, and any layer that is interested in it could subscribe to it. One practical implementation concept for such a publish/subscribe structure is a "blackboard", where each building block can "write down" values for a given name and where a building block can post a callback for a given name to be invoked when a value for this name is initialized or changes.

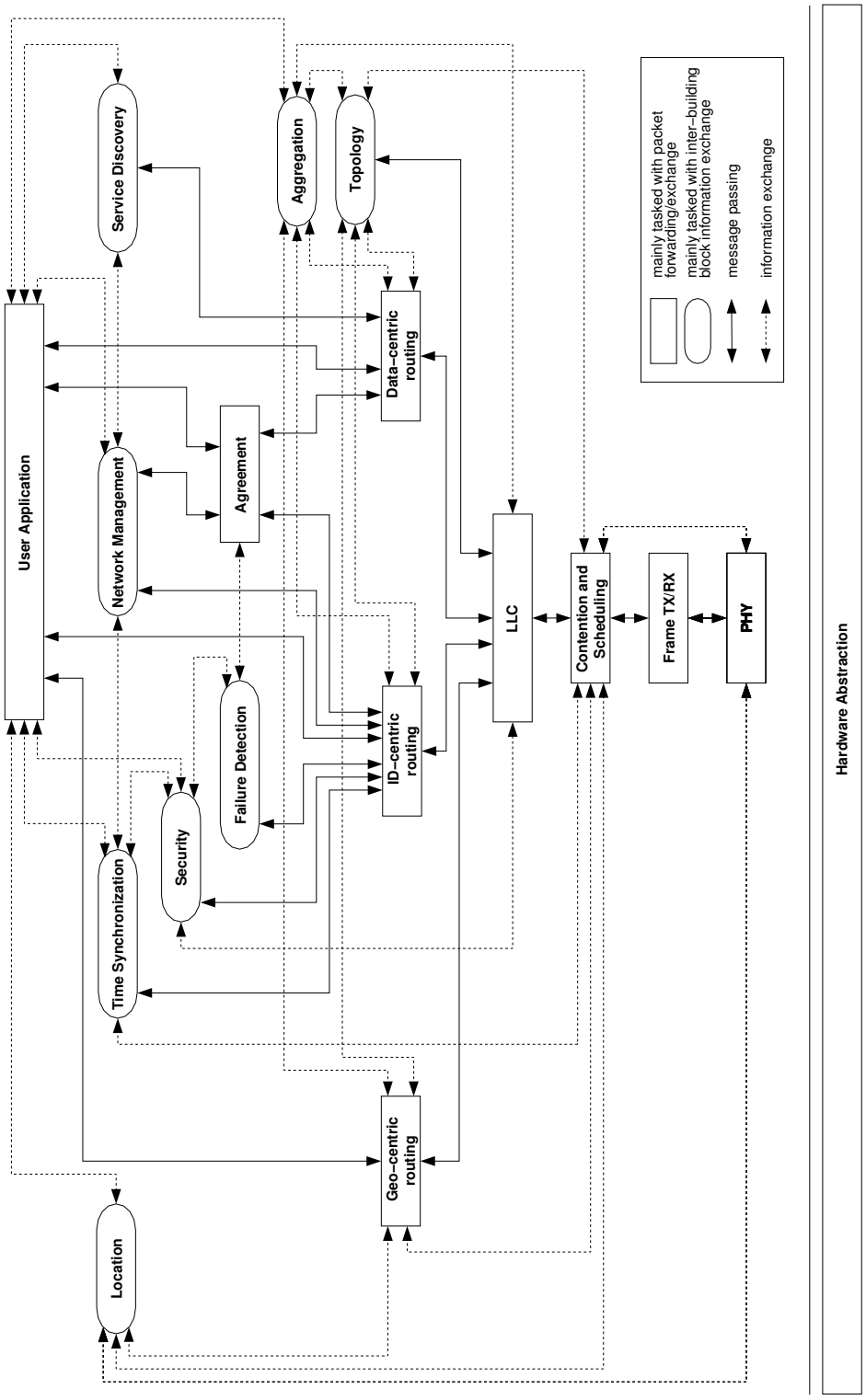

Figure 1: Possible structure of a WSN protocol suite

The advantage of such a publish/subscribe approach is the separation of individual building blocks, making the individual, separate development of protocol mechanisms easier.

Ideally, a protocol architecture should therefore be feasible that specifies names for such published information along with an appropriate packetfilter interface. In addition, some configuration language is necessary to describe which type of information a given building block relies upon and which therefore has to be present. In this way, a simple mixing and matching of individual building blocks from arbitrary sources and a comparison of different implementations of the same building block using different mechanisms should become easier.

3 Conclusion

The fragmentation of current WSN research is difficult to overcome and a complete harmonization of simulation and experimental environments is probably illusionary. Nevertheless, this paper has identified three approaches which could contribute to a better comparison and reuse of existing solutions: hardware abstractions for prototypes, characterizing the power consumption of actual hardware, and one possible approach to structure WSN protocol architectures that enables reusability in the face of the necessary tight integration of protocols in WSN.

Acknowledgements

Some of the content of this paper has been influenced by discussions with our colleagues from the EYES project, specifically, Thomas Lentsch, Michele Zorzi, and Paul Havinga; colleagues from FU Berlin, specifically Jochen Schiller and Hartmut Ritter; Adam Wolisz; and Andreas Willig. This work has in part been sponsored by the IST EYES project.

References

[1] The Network Simulator — ns-2. Project homepage at `http://www.isi.edu/nsnam/ns/`.

[2] EYES Project Website. `http://www.eyes.eu.org`, January 2003. Date of access.

[3] J. C. Mogul, R. F. Rashid, and M. J. Accetta. The Packet Filter: An Efficient Mechanism for User-level Network Code. In *Proc. 11th ACM Symp. on Operating System Principls*, Austin, TX, November 1987.