S2B2: Blackboard for Transparent Data and Control Access in Heterogeneous Sensing Systems

Michael Beigl, Monty Beuster, Daniel Röhr
Distributed and Ubiquitous Systems Group
TU - Braunschweig
Braunschweig, Germany
{beigl, beuster, roehr}@ibr.cs.tu-bs.de

Albert Krohn, Till Riedel, Christian Decker
Telecooperation Office (TecO)
TecO, University of Karlsruhe
Karlsruhe, Germany
{krohn, riedel, cdecker}@teco.edu

***Abstract*—This paper presents concept and first implementation of a Sensor System BlackBoard concept (S2B2). S2B2 is designed as a data store surrounded by software modules that operate on the Blackboard. The Blackboard is implemented as the only communication medium between modules within the sensor node. Modules can also transparently communicate with other modules even if they are hosted on a remote, different type of sensor node thus allowing true heterogeneity. The S2B2 module concept enables fine-grained decoupling of software functionality and separation of concerns. This allows rapid prototyping of functionality and simplifies real-time testing and debugging. The S2B2 provides the basis for integration of PC based and sensor node based applications. This paper introduces concept and architecture of S2B2 and shows the potential of the approach by presenting an example implementation. First results show that the system shortens programming time by at least 30%.**

Keywords: ubiquitous and pervasive computing, networked embedded sensor systems, middleware

I. INTRODUCTION

Networked sensing systems are often difficult to develop, maintain, debug, test and study. Sensor development requires programming of resource restricted network systems. This is very tedious due to long programming – downloading – testing circles, inferior tool support compared to PC programming and restricted in-situ debugging possibilities. Simulators as TOSSIM [1] or MiXiM [2] address the problem of programming and initial debugging by allowing programming of nodes in a simulated environment running on a PC. The drawback of such approaches is that real-world problems may not be appropriately transferable to a simulation model. Such a complex real-world behavior that cannot be modeled within the simulator may cause unexpected behavior from the nodes in real world experiments not visible in simulation. Debugging environments e.g. Sympathy and Emstar [3] address this issue by providing mixed environments where simulated applications and data from sensor nodes in real world can be incorporated. The approach requires a close integration of networked nodes into the simulation environment, especially a complete software representation of the behavior of a sensor node in the PC environment. The integration of physical sensor nodes is done on a per-node basis. Consequently the complete functionality of a sensor node can be either represented as a software simulation or integrated as a real-world node.

Based on our experiences with programming wireless sensor node applications we think that it would be beneficial to extend the integration of development support to a inner-node module level. Such an approach allows for a clear separation of concerns not only between the sensor nodes, but also between the various functionality of a sensor node. Our approach is module oriented. A clear defined set of functionality makes up one module that can communicate to other modules on the sensor network either locally or remotely, i.e. running as an internal sensor node module or as an external (e.g. PC based) module. The approach can be used for rapid prototyping of specific sensor node functionality without any need to care about the complexity of the rest of the node's functionality.

II. GENERAL CONCEPT

This paper will present a BlackBoard (BB)[4] oriented concept for Sensor Systems (S2B2) that uses a shared memory concept to allow applications access to software modules on sensor nodes. BB is a concept originating from Artificial Intelligence research. Many variations of the concept are known, but we restrict our introduction to a general overview. A Blackboard is a central data structure of a computing system that contains all knowledge of the system. Blackboard systems do not make a difference between the data and control flow – there is only knowledge in the system. Knowledge is produced and consumed by so-called Knowledge Sources (KS), which are often modeled as black box components that interpret the knowledge. KS are often the only legitimate active parts in a BB system. It is up to the KS to take actions according to a piece of knowledge – thus interpreting the knowledge as control structure. It should also be noted, that the BB is stateless – it only provides the access to the data. This way a BB decouples interpretation from communication – there is no binding nor implicit knowledge or state to be hold between the KS. Knowledge sources look up the BB to retrieve information without any knowledge where this information comes from. Thus, KS are not allowed to assume any behavior from other KS nor

should they communicate to other KS in other ways than via the Blackboard.
This may lead to performance problems, as e.g. any KS is required to scan the BB periodically to perform communication tasks (i.e. to receive information). To overcome this problem, some of the knowledge sources can be more closely coupled with the BB and are allowed to extend the communication mechanism of the BB. E.g. a subscribe/notify module is often used where other KS can subscribe to a type of message and get notified from the BB upon arrival. This way the subscribe/notify module extends the communication mechanism of the BB to events rather than only read/write/search. A subscribe mechanism requires that the structure of data stored in the BB has a known structure. Various types of such structures are known, most prominent are hierarchical structures [10] or typed data structures.

We introduced a Sensor-Systems BlackBoard S2B2 as an implementation of a BB on sensor nodes. S2B2 implements an extremely loosely coupled distributed BB. Due to the distribution aspect, the BB can have various views: local, nearby, regional or global. This is in contrast to other similar approaches, e.g. the Bulletin Board [9]. With S2B2 a global BB can be interpreted as a transformation of the real world covered by the set of BB, which is constructed through sharing the local BB views of all participating devices. The local BB is then the local perception or view of the real world. Such a view concept is very helpful for our RELATE application of the sensor system in that it supports the iterative processing of location knowledge. From our implementation of the BB it should be noted that in our model the BB is the only legitimate communication medium between modules.

Apart from the potential of designing and shifting scope of views, there are other practical benefits that can be addressed by implementing a BB on sensor nodes. Figure 1 depicts our BB concept: Each sensor node implements a Blackboard and several Knowledge Sources that we call modules. We have opt for the name module rather than KS to make clear that the work is not about usual topics as reasoning or learning. In Figure 1, 4 modules have access to the Blackboard of sensor node 1. Two of the modules (a,b) are implemented within the node, one is implemented outside the node on a PC (c) and one on sensor node 2 (module d). Due to the Blackboard coordinated architecture, module (c,d) are also seen as part of the system, the same as (a) and (b). In our approach, all parts of the sensor system including the basic operation of the node are modeled as modules. Thus our S2B2 model allows to run a sensor nodes functionality partially or completely by an external device, e.g. from a PC based system. In Figure 1, module (c) running on a PC may provide basic sensor operation functionality as sensor fusion remotely for Sensor Node 1.

The described approach has several advantages that are shortly listed.

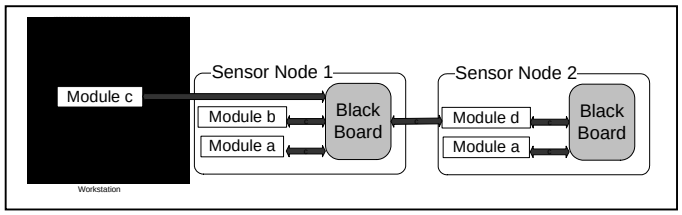

Figure 1. S2B2 Example of remote module access

A. Rapid prototyping and debugging with full PC based support. The BB approach allows a programmer to use existing, convenient PC based rapid prototyping tools and debugging capabilities. We will show at the end of the paper that this advantage could be quantified also in terms of development time. We are aware that the proposed approach may require a double implementation of an algorithm, first on the PC and then on to the embedded system. It is well known that the time required for debugging and testing a program is much higher compared to implementation time [7]. Thus the effect of lowering test&debug time outperforms the added implementation time.

B. Most appropriate programming language. Programmers may select the most appropriate development language for implementation from the full set of programming languages available on the PC.

C. Realistic operating conditions. Development of specific modules instead of complete applications outside a processing node lets these modules operate “virtually” in a more realistic environment. For example, modules that sample real-world sensor information are implemented within in the sensor node, but sensor processing is developed on the PC. This processing module is still able to operate on real-time, real-world data, in contrast to simulation environments.

D. Integration of development and simulation. The approach allows to integrate simulation, programming and testing. An algorithm developed on the simulator may be able to transparently use real-world data from sensing modules running on a sensor node. Vice versa a module operating on a sensor node may make use of the simulated data set on a PC. The approach provides the potential to easily build up standard test sets and environments.

E. Fine granular access to sensor node behavior. A characteristic of our S2B2 architecture is that the control and data flow can be easily tracked by logging data from the Blackboard. This gives access an extensive set of internal sensor node data that can be used for testing and debugging reasons.

III. RELATIVE IMPLEMENTATION

The system was implemented and tested in working environment in the context of the RELATE research project. The idea of the RELATE project is to provide

basic services for applications requiring peer-to-peer location information. The software architecture of the RELATE system should support application developers but also build the basis for research on P2P location systems using the RELATE platform. Various location modalities and different types of sensor node hardware are used within the project. Software modules for low-level sensor data reading as well as high level applications for the various RELATE platforms are implemented so far.

A. RELATE Blackboard Implementation Architecture.

The implementation of the RELATE Blackboard architecture (Fig. 2) consists of a set of Blackboard modules, the Blackboard and the underlying Operating System. This architecture is the same for all type of sensor modules thus allowing true heterogeneity. The Blackboard Implementation Architecture is an implementation of the S2B2 concept with a set of specialized modules and data structures that take into account the conditions of the resource restricted device the BB is implemented on. The OS also contains the AwareCon RF [5] communication protocol, a protocol build in firmware of the Particle transceiver boards [6].

The local Blackboard allows the integration of typed data using the ConCom [5] language. Modules can subscribe to a type and receive an event if the data is put on the BB from one of the nodes. This functionality is provided because of performance reasons and carried out by one of the System Modules (Fig. 2). Modules are grouped according to their functionality but are independent. The Location Modules contain all modules of the Location Stack as the Sensing and Postprocessing, Location Coordination, Region Manager for ad-hoc regional understanding of location of devices. The shift of the scope of the view of the surrounding "world" of location information is supported by the BB system. This allows us in the RELATE project to easily converge our understanding about the world – aka position of devices – from a very limited local view (local node/BB) to a small scope view (BB in vicinity) to a regional or even global view. System Modules provide basic functionality of the sensor node and the BB system. They enable access to various services on the nodes hardware. BB Operation modules care for operation task on the Blackboard. The Blackboard is also accessible from the Interfaces which are embedded into the (Particle) Operating System. The OS allows to implement functionality in a more non-modular manner that this is the case for a BB-only system. This can be seen as our tribute to the resource restriction of the sensor node platform underlying.

B. RELATE Blackboard Data Structure, Communication Modalities and operations.

The Interfaces part also contains the wrapper functionality that allows a transparent access to the Blackboard from external modules and applications. Such Interfaces enable applications and devices from outside to access the sensor node's local Blackboard transparently as if they run locally. There are two interfaces provided: The **Console** interface is a high-level text based interface to the Blackboard. This interface is mainly used to enable a human user to retrieve and change information on the Blackboard, but could also be used to for writing script programs both on the PC or embedded node thus allowing rapid prototyping. A more compact and machine oriented access is possible with the **AwareCon** [5] interface, which exposes the same functionality as the console but in a byte encoded opcode format based on the ConCom protocol.

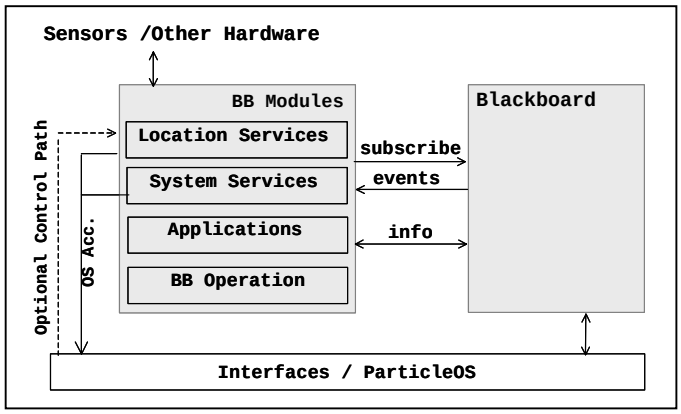

Figure 2. Exemplary Blackboard Implementation Architecture

Thus, S2B2 provides 3 modalities to access Blackboard information: Internal APIs on a sensor node, text based commands via a Console and ConCom packet format. All three are different communication modalities providing the same operations that can be performed on S2B2. Operations are performed on the information stored in the Blackboard which are structured as list of typed tuples following the ConCom [5] specification. Operations on the BB are:

- Read/Filter read: Read a single tuple, a set of tuples (according to a specified type pattern) from the BB.
- Write: Write a tuple to the BB. A scope parameter indicates if the write is intended to go the local BB or the region etc. Writing to the regional BB causes communication of the tuple via the network.
- Subscribe: Subscribe to one or more tuple-types to receive an event upon arrival.
- Delete/Filter delete: Delete a single tuple, a set of tuples (according to a specified type pattern) from the BB.

C. RELATE Console Operation.

Operation on the local RELATE Blackboard do not require that all modules described above are running within the embedded software of a RELATE node. With the concept of interfaces and especially the Console concept transparent remote access to the RELATE node is possible. The Console exposes a simple minimal text based interface to the Blackboard using a standard input/output command shell. This minimal interface can be extended by combining

the minimal interfaces commands to write more powerful scripts using e.g. shell commands.

Console system thus simplifies development of programs by letting a programmer first develop and test parts of a Blackboard module (e.g. an application program) on the PC by accessing the RELATE sensor nodes blackboard information storage remotely using the console. Standard I/O can also be used to pipe BB information into a file helping to store and retrieve location information for evaluation or other purposes.

IV. First Results

The implementation of S2B2 was introduced as a development, testing and field study platform for the RELATE project replacing the existing "standard" embedded sensor node development and testing process. We collected about 2 month experience with S2B2 so far. Still, many of these experience are more quantitative rather than qualitative, but the benefit of separating concerns and thus enabling a more decouple development process is difficult to measure.

From a quantitative perspective it is important to note, that the S2B2 approach provides various possibilities for improving the development process. E.g. a physical measurement component could be replaced by a simulated or play-backed reality thus allowing to generate a more comparable set of results. On the other hand, a location detection algorithm developed based on a simulator can be directly tested on real-world measurements taken at sensor nodes by simply replacing the sensing & post processing module in the simulation and forward control and dataflow to the module to the remote Blackboards of real sensor nodes.

There are also several quantitative results to be reported from the first testing phase. In this first testing phase we concentrated on the implementation of small software modules as simple location algorithms and application programs for user output[1]. Table 1 shows first results of this testing phase. It can be seen that the performance of development time increases at about 30%, not considering the time for flashing the embedded controller which saves another 46 seconds flashing time for each round of programming in average. The parameters do also not reflect the general positive effect of speed-up in implementation due to the more clear separation of concerns.

The speed-up in programming time goes along with a decrease in the lines of code (LOC) required for writing the programs (almost 50%) which is also an indicator for the simplified development process.

TABLE I. Development Performance Parameters

Parameter	Value
Dev-time Cons. (min)	Max 10, Min 7, Avg 8
LOC Console	Max 26, Min 25, Avg 26
Dev-time embed (min)	Max 15, Min 10, Avg 11
LOC embedded	Max 64, Min 34, Avg 50
Flashing time	Max 56s, Min 32s, Avg 46s

V. Conclusion

This paper introduced the concept of using a Blackboard-like architecture for Sensing Systems. We have shown some initial quantitative and qualitative advantages for developing and testing applications for sensing systems. Next steps will require more extensive studies on the improvement in the development process, especially in the context of complex software projects.

So far we only compared development time neglecting the effect of improved development process due to clearer separation of concern, more flexibility in combining functionality and a higher reuse-factor. We will research these parameters in the future and also seek to quantify these effects. Also, we see great potential in using the Blackboard architecture and the Console as a flexible inter-sensor node system for operation in highly heterogeneous environments using various brands of sensor nodes but especially using sensor nodes together with other mobile devices and together with stationary – e.g. environment installed – devices. The potential here is that relocation of services can be ad-hoc and easily performed using such a method.

[1] Conditions: Programming of a microprocessor PIC Microchip PIC 18LF series under C (embedded programming) using CCS C. Programming Console using Bash shell. Flashing uses CCS ICD with/without verify on. Programming and Flashing performed on several Pentium M/IV PCs from 1.5GHz to 2GHz. Note: Flashing and C-Programming is performed using the fastest compiler and Flash-programmer available for the used microprocessor; measured times would be substantially longer when using other compilers and programmers.

REFERENCES

[1] P. Levis et al. "TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications," Proc. 1st ACM Conference on Embedded Networked Sensor Systems (SenSys 2003), November 2003.

[2] Otto Visser, MiXiM, Master's Thesis, TU Delft, NL, August 2005.

[3] N. Ramanathan, E. Kohler, D. Estrin. Towards a Debugging System for Sensor Networks, International Journal for Network Management, 2005.

[4] R. Englemore, T.Morgan. Blackboard Systems, Pub. Addison-Wesley Pub. Co. 1998

[5] Beigl, M., Krohn, A., Zimmer, T., Decker, C., Robinson, P.: AwareCon: Situation Aware Context Communication. Proceedings of Ubicomp 2003, Oct. 12-15, Seattle, USA

[6] Particle Computer. http://www.particle-computer.net

[7] Fairley, R. 1985 Software Engineering Concepts. McGraw-Hill, Inc

[8] Albert Krohn, Mike Hazas, and Michael Beigl, Removing Systematic Error in Node Localization Using Scalable Data Fusion, EWSN 2007, to appear.

[9] Lifton, J., Seetharam D., Broxton, M., Paradiso, J.: Pushpin Computing System Overview: a Platform for Distributed, Embedded, Ubiquitous Sensor Networks. Pervasive 2002, Proceedings of the Pervasive Computing Conference, Zurich Switzerland, 26-28 August 2002, Springer Verlag, Berlin Heidelberg, pp. 139-151

[10] Weiss, M., Stetter, F. A hierarchical blackboard architecture for distributed AI systems, Software Engineering and Knowledge Engineering, 1992. Proceedings., Fourth International Conference on Volume , Issue , 15-20 Jun 1992 Page(s):349 - 355