The Unified Framework for Sensor Networks: A Systems Approach

Geoff V. Merrett, Alex S. Weddell, Nick R. Harris, Neil M. White, and Bashir M. Al-Hashimi

School of Electronics and Computer Science, University of Southampton, UK
{gm04r, asw05r, nrh, nmw, bmah}@ecs.soton.ac.uk

Abstract – Since its introduction in the mid-1970s, the OSI Basic Reference Model (OSI-BRM) has been widely used as a foundation for communication models and standards. While many of these have modified the OSI-BRM for specific communication requirements (protocols such as ZigBee and Fieldbus – used in sensor networks), little structure or standardisation has been developed for other aspects of the hardware/software interface – for example sensing, energy management, actuation or locationing. Such processing is often implemented in the application layer of the communications stack, resulting in an unstructured, top-heavy and confusing stack. Alternatively, processing is performed off-chip or in separate unstructured software. In this paper, we propose the Unified Framework for the structured design and implementation of multiple interfaces on a sensor node. The framework creates unified stacks by connecting individual stacks (containing distinct functionality) via a shared application layer. We present the application of the framework to create a unified stack, structuring both communications and sensing. The process of extending a unified stack for implementing energy management, locationing and actuation is also discussed. The proposed framework establishes a structured platform for the formal design, specification and implementation of sensor and wireless sensor networks.

I. Introduction

Communication protocol models have been utilised for decades, aiming to formalise, structure and provide interoperability between the tasks of the networking system. The OSI Basic Reference Model (OSI-BRM) [1] was introduced in the mid-1970s, proposing a basic layered architecture for communication protocols – as shown in figure 1. The primary functions of each layer of the OSI-BRM are outlined below:

- **Physical Layer:** performs channel coding, bit-level communication over the physical/wireless medium, and directly controls the communication hardware.
- **Data link Layer:** consists of two sublayers: logical link control (for example frame and error control) and medium access control (controlling multiple accesses to a shared communications medium).
- **Network Layer:** provides network related protocols such as packet routing.
- **Transport Layer:** provides high-level end-to-end reliability, such as flow control, connection management and error recovery.
- **Session Layer:** manages sessions between networked devices (this layer is rarely implemented, with functionality absorbed by the application layer).
- **Presentation Layer:** manages the syntax and semantics of communicated data (this layer is rarely implemented, with functionality absorbed by the application layer).
- **Application Layer:** contains a range of commonly used high-level protocols. In smaller devices, the entire application may reside within the application layer.

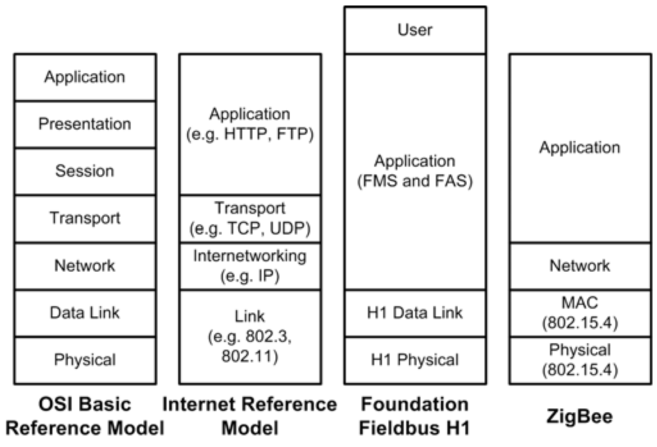

Figure 1 – The OSI-BRM [1], IRM [2], Foundation Fieldbus H1 [3] and ZigBee [4] communication stacks.

Many widely used communication stacks and models (for example the Internet Reference Model (IRM) [2] shown in figure 1) have added, removed and merged layers from the OSI-BRM to tailor the model to their requirements. However, the OSI-BRM remains an important building block for modern communication protocols.

Fieldbus H1 [3] was designed as a network architecture for process control applications, such as sensor networks. It can be seen from figure 1 that the majority of the higher layers have been removed, as the functionality provided by these layers (for example packet routing, flow control and connection management) is not required by the network. Fieldbus adds a User layer to the top of the protocol stack, which allows additional functionality to be provided, such as a user interface.

The ZigBee specification [4] defines a low-cost, low-power wireless communication standard that is particularly suited to wireless sensor networks. The ZigBee protocol stack (shown in figure 1) uses the IEEE 802.15.4 [5] Physical and Medium Access Control (MAC) layers. The transport layer is omitted, with the relevant functionality being absorbed by neighbouring layers.

The OSI-BRM was designed to represent only the communications functionality of a networked node. However, ZigBee (based on the OSI-BRM) uses the communications stack to contain software for the entire device, placed in the large and subdivided application layer. Some Fieldbus networks also place the program in the application layer, or alternatively use off-stack software that does not promote reuse or interoperability. This was found to be true in all of the models and stacks investigated.

The TinyOS operating system [6] (used in many wireless sensor nodes, such as the Crossbow motes [7]) has a layered software stack, where data from the sensors (and any other hardware) is directly processed in the application layer [8]. Therefore, any processing of sensory data must also take place within the application layer. In energy harvesting nodes such as Heliomote and Prometheus [9, 10] the application must monitor voltage levels and incorporate cursory temperature compensation. In these nodes, such energy management operations occur in the application layer. While communication gets formally specified in the majority of implementations, other interfaces (such as sensing and energy management) are left unstructured.

In this paper, we introduce the Unified Framework to formally specify multiple interfaces on a sensor node. The framework creates a unified stack by connecting individual stacks (containing distinct functionality) through a shared application layer. This establishes a structured platform for the formal design, specification and implementation of modern sensor and wireless sensor nodes.

II. The Unified Framework

In the previous section it has been shown that, while the communications interface is formally structured, other interfaces (such as sensor processing) are generally unspecified and often integrated into an unstructured and overflowing application layer. This is not ideal for sensor processing, which is arguably as important as communications to the functionality of a sensor node.

We propose the Unified Framework, which specifies and structures multiple interfaces on a sensor node. The framework provides a basic template stack, based upon the OSI-BRM, from which different interface stacks can be derived. A number of these stacks with distinct functionality are combined via a shared application layer, forming a unified stack. The Unified Framework's basic template stack is shown in figure 2. Stacks developed following the framework are derived from this template. The layer boundaries are placed to accommodate a wide range of hardware interfaces, and to allow different protocols to be interchanged without redefining surrounding layers. The functions of the layers are:

- **Interface Layer:** directly interfaces with the relevant hardware, and hides the complexity of the circuitry from the higher layers. For communications, this layer represents the physical layer of the OSI-BRM.
- **Medium Layer:** provides low-level processing, and hides the complexity of interfacing with the medium (for example, the wireless channel or the sensed phenomenon) from the higher layers. For communications, this layer represents the data link layer of the OSI-BRM.
- **Management Layer:** provides high-level functionality, and hides the complexity of groups of objects (for example a network of nodes, or number of different energy sources) from the higher layers. For communications, this represents both the network and transport layers of the OSI-BRM.
- **Shared Application Layer:** contains cooperative functionality that passes data between individual stacks. For communications, this layer represents the application layer in the OSI-BRM.
- **Program Area:** contains the end-user's application. The Unified Framework recommends against the use of the program area wherever possible, and instead suggests that functionality is transferred to a relevant stack. Naturally, there are some implementations where this may not be feasible, and so the program area is incorporated into the framework for these situations.

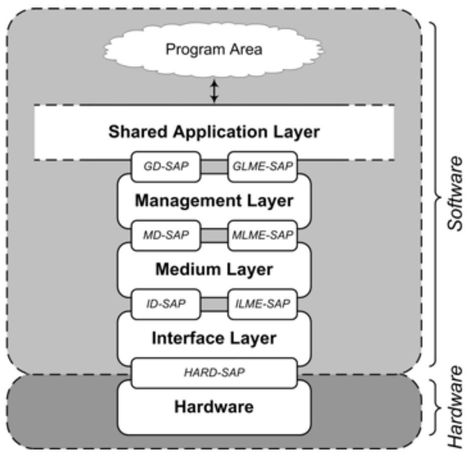

Figure 2 – The Unified Framework's basic template stack.

Communication between neighbouring layers takes place through the service access points (SAPs). At each layer boundary, there are two SAPs – the data SAP (xD-SAP), used for transferring data between layers, and the layer management entity SAP (xLME-SAP), used for layer management functions such as getting and setting layer parameters. The physical layer interfaces with hardware through the HARD-SAP; the services and connections of which are specific to individual implementations.

Communications and sensing are fundamental parts of any sensor node. Figure 3 shows an example of a unified stack (created following the Unified Framework) designed to implement both the communication and sensing interfaces.

The communication stack is similar to that of ZigBee [4]. The physical layer interacts directly with the communication hardware, and hides the complexities of the communication circuitry from the MAC layer. The MAC layer provides features including channel access (redundant in non-broadcast networks), frame management, low-level error detection and security, and hides the complexities of the communication medium from the network layer. The network layer controls message routing and subnet formation and maintenance, thus providing the shared application layer with a view of the network as a single entity.

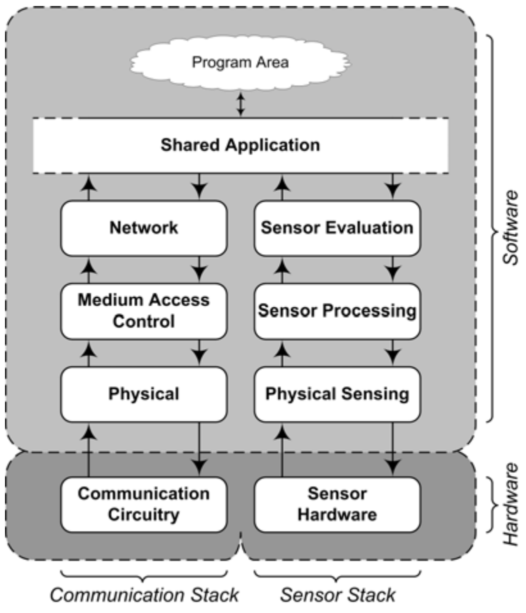

Figure 3 – A unified stack incorporating both communication and sensing interfaces.

The sensor stack implements the intelligent sensing functionality of a node: interfacing with the physical sensing circuitry, processing sensory data, and assessing its reliability. The desired output data from an intelligent sensing system includes an estimate of the measurand and its uncertainty level. Estimates of error and uncertainty are useful in a multi-sensor fusion context [11], so that greater attention can be paid to accurate data. The software system must interface with the physical sensor, and any physical pre-processing and conversion stages. The sensor stack (shown in figure 3) is split into three layers:

- **Physical Sensing Layer:** interfaces with the sensor hardware by energising sensors and obtaining readings. It provides raw sensor data to the layer above, but hides the complexities of interfacing with the sensor and conversion circuitry. For example, it switches a pressure sensing device on and obtains a reading from the ADC, providing raw sensor data to the processing layer.
- **Sensor Processing Layer:** accepts the raw sensor data from the layer below, and performs preliminary processing. It provides the layer above with adjusted data, and can perform localised data fusion. In the context of a pressure measurement system, it linearises the data (through use of a look-up table) and provides temperature compensation by fusing pressure data with temperature information.
- **Sensor Evaluation Layer:** interfaces with the physical sensing layer and refines data through comparison with the sensor model. It provides the shared application layer with an adjusted sensor reading, complete with error bars and indications of sensor faults. The stage takes into account the age of the sensor and any previous damaging events, in line with the sensor model and condition monitoring and fault detection logic, to estimate drift and uncertainty.

This structured approach to sensor interfacing brings with it a number of advantages. As with the communication stack, complexities of lower levels are hidden from those above. Layers may be redefined to provide differing functionality without affecting the operation of neighbouring layers. In addition, layer operation may be reconfigured through the management SAPs.

The shared application layer handles collaboration between the individual stacks – passing high-level data between them, under self-guidance or guidance from the program area. For example, a packet may be received via the communication stack requesting a measurement. The shared application layer would then request this data from the sensor stack and pass the result back to the communication stack for transmission. This layer also provides the interface with the program area (if implemented), offering common protocols and access to the individual stacks.

III. Using the Framework to Extend a Unified Stack

The nodes in a wireless sensor network are typically resource-constrained, relying on batteries or energy harvesting to supply their energy. Where multiple energy sources are used, the management of the energy subsystem can become a complex process, with the flow of charge between sources requiring intricate control.

By partitioning the energy management process into distinct tasks, the implementation can be migrated from the program area to a separate stack, removing a source of complexity and reducing the volume of unstructured functionality. A unified stack – such as that discussed in the previous section – can be extended (following the Unified Framework) to include an energy management stack (as shown in figure 4). Here, the energy management process is divided into three layers:

- **Physical Energy Layer:** obtains information about voltages of energy stores (indicating the amount of energy remaining), monitors the yields from energy harvesting, and controls physical switching in order to direct the flow of energy.
- **Energy Analysis Layer:** takes data from the physical energy layer and for each source provides information such as the rate of energy usage, energy generation, and stored energy (through use of source models).
- **Energy Control Layer:** takes a high-level view of the energy subsystem – making decisions about energy sources, switching, and the general flow of energy. It indicates to the shared application layer the residual energy, and the sustainability of present usage.

To illustrate the operation of the energy stack, a node utilising vibration harvesting, a capacitor and a rechargeable battery is considered. The vibration harvesting device acts to supply energy, the capacitor acts as a primary buffer, and the rechargeable battery as a secondary buffer. The physical energy layer hides the complexities of interfacing with the physical hardware. It controls switching, allowing charging and energy transfer operations to take place in line with directions from higher layers. It monitors the voltages of energy sources and the amount of energy being harvested, passing values to the energy analysis layer. The energy analysis layer takes this physical data and uses source models to derive the amount of energy stored and its rate of generation (for example, taking account of temperature effects and temporal deterioration to derive the real amount of energy stored in the battery). It accepts commands from the layer above regarding energy distribution, and translates these into switching commands for the physical layer to execute. The energy control layer takes the refined information about energy sources and calculates whether the rate of energy usage exceeds the rate of generation. Such information is sent to the next higher layer. The layer makes decisions regarding energy transfer and distribution, which are sent to lower layers. Rechargeable batteries are generally limited to 300-500 recharge cycles, so energy transfer operations must be minimised to prolong the lifetime of the platform [9].

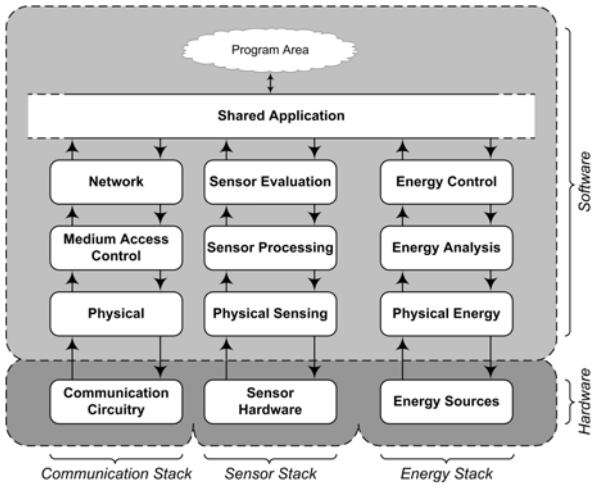

Figure 4 – A unified stack incorporating communications, sensing and energy management interfaces.

Additional extensions to the framework could include actuation and locationing stacks. Each application of location-aware computing treats location tasks differently [12]. However, recent convergence in location technology trends permits the formation of a generalised location stack. An actuation stack would permit actuation commands to be expressed via the shared application layer as a high-level command, which could be translated by the stack into low-level control of operations. It can be seen that these extensions could be incorporated into a unified stack.

Further extensions to the concept of the unified stack include the creation of software/software interface stacks, where data present in the shared application layer can be processed in the same fashion as data obtained from hardware. An example of this could be in performing data fusion using packets received from other nodes in the network. Functionality can be migrated from the program area into a distinct, structured stack. However, this is outside the scope of this paper.

IV. Conclusions

In this paper, we have proposed the Unified Framework – a means for specifying and structuring the multiple interfaces on a sensor node. The framework provides a basic template stack from which different interface stacks can be derived. A number of these stacks with distinct functionality are connected via a shared application layer to form a unified stack. We have presented a unified stack for communication and sensing interfaces, and then shown how it can be extended following the Unified Framework to implement additional functionality such as energy management. The flexibility of the framework permits the structuring of other interfaces, for example actuation and locationing. The framework establishes a structured platform for the formal design, specification and implementation of the nodes in sensor and wireless sensor networks.

References

[1] ITU-T X.200, "Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model", 1994

[2] D. Meyer and G. Zobrist, "TCP/IP versus OSI," *IEEE Potentials*, vol. 9, pp. 16-19, 1990.

[3] S. Kolla, D. Border, and E. Mayer, "Fieldbus networks for control system implementations," *Proc. Electrical Insulation Conf. & Electrical Manufacturing & Coil Winding Technology Conf.*, pp. 493-498, Sep. 2003

[4] ZigBee Alliance Document 053474r06v1.0, "ZigBee Specification", 2004

[5] 802.15.4™-2003, "IEEE Standard for Information Technology - Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)", 2003

[6] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," *Proc. 1st Int'l Conf. Embedded Networked Sensor Systems (SenSys'03)*, Los Angeles, CA, pp. 126-137, Nov. 2003

[7] Crossbow Technology Inc. (San Jose, CA), www.xbow.com; last accessed Jun. 2005.

[8] D. Patnode, J. Dunne, A. Malinowski, and D. Schertz, "WISENET - TinyOS based wireless network of sensors," *Proc. IECON'03. 29th Annual Conference of the IEEE Industrial Electronics Society, 2-6 Nov. 2003*, Roanoke, VA, USA, vol. Vol.3, pp. 2363-2368, Nov. 2003

[9] X. Jiang, J. Polastre, and D. Culler, "Perpetual Environmentally Powered Sensor Networks," *Proc. 4th Int'l Conf. Information Processing in Sensor Networks/Special Track on Platform Tools and Design Methods for Network Embedded Sensors (IPSN/SPOTS)*, Los Angeles, CA, Apr. 2005

[10] A. Kansal, D. Potter, and M.B. Srivastava, "Performance aware tasking for environmentally powered sensor networks," *Proc. SIGMETRICS'04/Performance: Joint Int'l Conf. Measurement and Modeling of Computer Systems*, New York, NY, vol. 32, pp. 223-234, Jun. 2004

[11] P.J. Boltryk, C.J. Harris, and N.M. White, "Intelligent sensors-a generic software approach," *Journal of Physics: Conference Series*, vol. 15, pp. 155-60, 2005.

[12] J. Hightower, B. Brumitt, and G. Borriello, "The location stack: a layered model for location in ubiquitous computing," *Proc. Proceedings Fourth IEEE Workshop on Mobile Computing Systems and Applications, 20-21 June 2002*, Callicoon, NY, USA, pp. 22-8 BN - 0 7695 1647 5, Jun. 2002