UPS: Unified Protocol Stack for Wireless Sensor Networks (Extended Abstract)

Chen-Hsiang Feng, Ilker Demirkol and Wendi B. Heinzelman
Department of Electrical and Computer Engineering
University of Rochester, Rochester, NY, USA
Email:{feng,demirkol,wheinzel}@ece.rochester.edu

I. Introduction

The complexity of allowing different protocols to operate in concert in the same layer has generally been ignored in wireless sensor networks. In this paper, we propose a complete protocol stack framework called UPS (Unified Protocol Stack) as a solution for incorporating different protocols within WSNs through enabling the co-existence of multiple modules in same stack layer as well as providing unified access to cross-layer data. The system block diagram for UPS is shown in Fig. 1.

The use of a common interface of packet switching enables the modules in the same layer and different layers to seamlessly work together. Furthermore, we refine the concept of centralized information sharing services by separating the concepts of "information" and "service". Unlike previous approaches using a centralized unit outside the protocol stack or integrated within the protocols themselves, services should be self-sufficient protocol stack modules working side-by-side with other stack modules. Moreover, information should be retrieved from service modules through a unified interface.

We demonstrate the gain of this novel approach by comparing it with previous OSI stack designs with a single protocol at each layer through both implementation on Tmote Sky motes and simulation using the TOSSIM simulator. We implemented the XLM [1] cross-layer WSN protocol as well as a network layer multicast protocol (called RBMulticast [2]) as protocol modules in the network layer, showing that the network layer traffic co-exists and shares the same MAC layer without extra overhead. Our results show that utilizing UPS with both XLM and the multicast protocol running simultaneously, we can reduce the network traffic by up to 20% compared with a network that only runs XLM and by up to 42% compared with a network that only runs the multicast protocol.

II. UPS Framework

UPS (Unified Protocol Stack) consists of groups of *interfaces* for interconnecting protocol modules and for enabling information sharing among the protocol modules. The following sections describe the interfaces.

This work was supported in part by the National Science Foundation under grant # CNS-0448046 and in part by a Young Investigator grant from the Office of Naval Research, # N00014-05-1-0626.

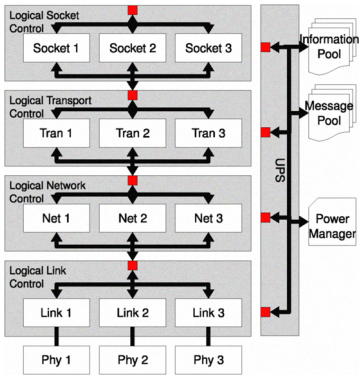


Fig. 1. A high level system block diagram of the UPS framework.

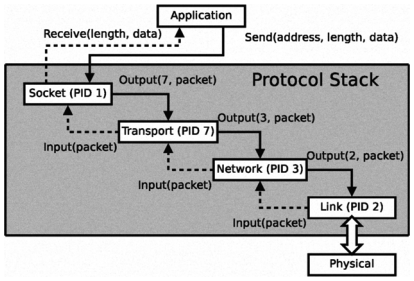


Fig. 2. Packet flow diagram for the UPS framework.

A. Logical Control (LC)

Logical Control is a generalized extension of Logical Link Control in the IEEE 802 family of standards. Its purpose is to multiplex packets passed from the upper layer (when transmitting) and demultiplex packets from the lower layer (when receiving). This is the key to co-existence of multiple protocols in the same layer. In UPS, the first byte of the packet header of each layer is reserved for *Logical Control* packet switching. The function calls for the *Logical Controls* interfaces are as follow:

```
Input(Packet);
Output(Lower layer Protocol ID, Packet);
```

We demonstrate the use of these interfaces using a schematic diagram in Fig. 2. The *Output* interface requires the Protocol ID of the next (lower) layer module for multiplexing of the output packets from different modules. The `Input` interface does not need the Protocol ID parameter because it is indicated by the leading byte of the packet header (the upper layer protocol module's packet header).

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.6991
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.6991

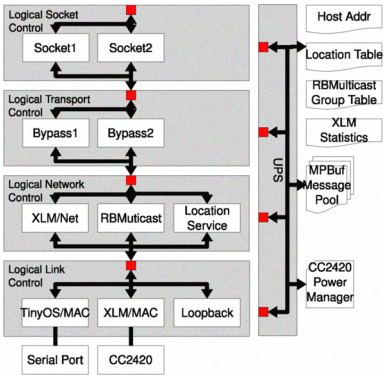


Fig. 3. Block diagram of the UPS framework developed in this paper.

B. Message Pool

Even if protocol stack modules conform to the same *Input* and *Output* sending interfaces, different stack modules still need a standardized internal packet format in order to support packet switching and module reuse. A *Message Pool*, which is basically a "pool" of available memory segments, provides an interface for accessing dynamic memory storage. This enables all modules to access packets of the *same format*. The pseudo-code of the *Message Pool* interface is as follow:

```
Memory Block = get(): get a memory block from the
                      pool.
put(Memory Block): return Memory Block to the pool.
```

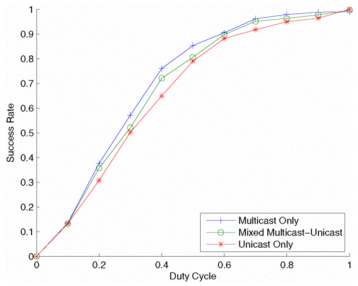
C. Information Pool

UPS avoids centralized information storage by not providing storage explicitly. Instead, UPS assumes that the responsibility of information storage is on the provider side, and thus each module independently manages its own information stores. For example, our Location Service module implementation has a location lookup hash table, which can be accessed by any module through the following pseudo-code:

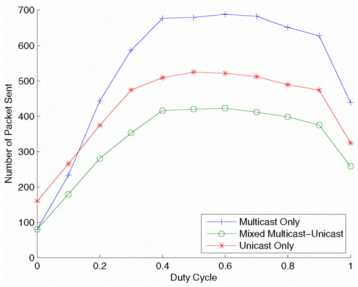
```
access( InformationID, Operation, Methods, Argument),
   where:
   InformationID: ID of the Location Service module
   Operation: Get or Set
   Methods: Location, Distance, Add or Remove Node...
   Argument: a memory space that has a node address
             as the first two bytes and space for a
             return value following the node address
```

III. UPS Experimental Results and Evaluation

Fig. 3 shows the block diagram of the system we developed to highlight the benefits of UPS. XLM/Net and XLM/MAC are the Network layer and Link layer modules extracted from the cross-layer protocol XLM [1], which is a hand-shake receiver-based unicast protocol. RBMulticast [2] is a network layer multicast protocol that also makes use of XLM/MAC to provide multicast communication. RBMulticast stores information about the multicast members in an RBMulticast Group Table, which is accessible via the information pool *access* interface. *MPBuf Message Pool* provides dynamic memory access to packet storage space for all stack protocols.



(a) Success rate.



(b) Total number of packets sent.

Fig. 4. Statistics of the experimental network in different scenarios.

We assume three scenarios. In the first scenario, motes can only run the RBMulticast protocol, and hence unicast communication must be done using the multicast communication. In the second case, motes can run both the unicast and the multicast protocols using UPS. In the last case, motes can only run the unicast protocol, and multicasting is done by consecutive unicast transmissions.

The success rate of all three scenarios, shown in Fig. 4(a), only differ slightly, supporting our claim that protocols that share modules will behave similarly. That is, all three cases reveal the same characteristics of the XLM/MAC layer protocol.

The total number of packets sent are shown in Fig. 4(b). The results here also clearly show the benefits of using a mixed protocol approach in that the multicast only case has almost twice the number of sent packets compared with the other two cases. The UPS-enabled mixed multicast-unicast case provides the same functionality with fewer packets sent and hence achieves better energy efficiency.

Our future work will focus on the evaluation of UPS for hybrid wireless networks with multiple radios and multiple MAC protocols. New routing protocols can be developed within the UPS framework for more efficient networking, benefiting from the support of multiple MAC protocols. Future updates of this work will be available online at http://www.ece.rochester.edu/research/wcng/.

References

[1] Akyildiz, I., Vuran, M., Akan, O.: A cross-layer protocol for wireless sensor networks. In: Proc. of CISS 2006. (March 2006)

[2] Feng, C.H., Heinzelman, W.B.: Rbmulticast: Receiver based multicast for wireless sensor networks. IEEE Wireless Communications and Networking Conference (WCNC '09), April 2009.

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.6991
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.6991