An Information-sharing Architecture for Wireless Sensor Networks

Christophe J. Merlin and Wendi B. Heinzelman
Department of Electrical and Computer Engineering,
University of Rochester, Rochester NY
{*merlin, wheinzel*}*@ece.rochester.edu*

***Abstract*— Recent work on cross-layer schemes have demonstrated the need for a unifying wireless sensor networks architecture that provides more integration than the standard layered OSI protocol stack yet is flexible enough to support different applications. In this paper, we propose a new information-sharing architecture for sensor networks that can support existing protocols while simultaneously providing a platform for advanced cross-layer improvements. Our new architecture utilizes different services and data structures for providing information that can be shared among all layers of the protocol stack for increased network performance. This architecture has the advantage of maintaining the existing OSI layer structure while enhancing the performance of the network by providing a common framework for each protocol in the stack to access necessary information for protocol optimization.**

I. Introduction

AODV, DSR, PEAS, *i3*, rumor routing, IEEE 802.11, DAPR, SMAC, directed diffusion... Researchers have made numerous contributions to the wireless sensor network and ad-hoc network fields in only a few years. While there is much more work yet to be done, further thought needs to be given to a federating architecture that can both support and enhance the performance of these protocols.

Cross-layer schemes have advanced the idea that two or more layers can benefit from the same information; for instance, different decisions might be taken at the routing and node activation levels based on the distance of the sensor to its next-hop neighbor. Similarly, the physical layer may wish to know this information in order to transmit a packet to the next-hop sensor with only the necessary power to reach it.

There are two main types of cross-layer improvements:

- *Information Sharing*: several layers share information.
- *Layer Fusion*: operations from two or more layers are conducted jointly to optimize their output.

In previous work [1], we showed that while the former can be beneficial, the latter shows suprisingly little improvement in the face of other design optimizations. We therefore propose a platform that provides support for *Information Sharing*, while still leaving open the possibility for *Layer Fusion*.

This paper is organized as follows. Sections II and III offer detailed aspects of our new architecture and how to manage its local data stores. Section IV provides related work, and Section V concludes this paper.

This work was supported in part by NSF # CNS-0448046.

II. A New Architecture for Cross-Layer Information Sharing

There are numerous cross-layer improvements that have been proposed to improve sensor network protocol performance. For example, at the physical layer, a node can use the distance to the next hop in order to utilize only the power necessary to reach it—and consequently contribute to lower contention in the network. This requires information from higher layers of the protocol stack, namely the distance to the packet's destination node.

Similarly, at the MAC layer, various decisions can be made about the timing of a transmission depending on the criticality of the data contained in a packet, which must be obtained from the application layer. At the routing level, a protocol may elect a different route based on the contention of its next-hop, or it may favor routes that utilize sensors that are less important to the sensing application and thus more dispensible. At the node activation layer, the remaining energy combined with the criticality of a node's data may prompt a sensor to stay off.

All of these cross-layer optimizations share a common theme—information must be shared between the layers to appropriately optimize certain parameters of the individual protocols.

A. The Need for a New Architecture

Our goal in creating this new architecture is to support the exchange of fundamental information that is beneficial to all stacks, and at the same time, to create an architecture that is compatible with existing and future protocols, both layered and cross-layer. To this end, we have attempted to identify a basic list of important parameters necessary for improving the performance of many protocols. These include:

- Node location.
- Node remaining energy (ϵ_{rem}).
- Compute resources such as CPU load, RAM use, and remaining storage capacity.
- Compute and sensing abilities such as compression, aggregation, error correcting abilities, and a set of variables a sensor can monitor.
- Contention around the node.
- SNR or probability of error (P_e), reflecting the quality of the link to any neighbor.

These node and network features may be required by one or more protocols, and oftentimes they are not straightforward

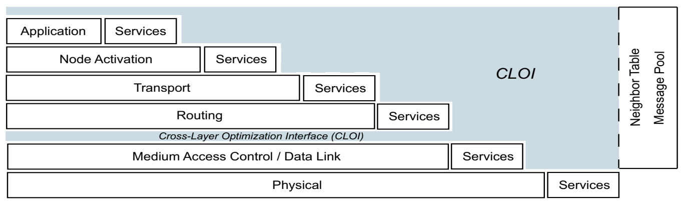

Fig. 1. An information-sharing sensor network architecture for cross-layer optimizations.

to obtain. For example, while determining a node's compute resources may be simple, determining current link SNR or contention around a node requires specialized *services* to find this important and time-varying information. In current architectures, the individual protocols are responsible for obtaining such information, and oftentimes these services are replicated in different layers, or a cross-layer design is used so that the information is readily available to multiple protocols in the stack. We believe a better architecture is one that provides a common framework for obtaining such information and enabling all protocols in the stack access to it.

B. A New Unifying Architecture

We propose the new wireless sensor network architecture shown in Figure 1. Our architecture retains the layered structure such that each layer is matched to a communication function in order to maintain a practical and simple design. The layers are: Physical, Medium Access Control (MAC)/Data Link, *CLOI* or *Cross-Layer Optimization Interface*, Routing, Transport, Node Activation, and Application. Other functions useful to the global communication scheme can be linked to services with a specific position in the protocol stack.

The Cross-Layer Optimization Interface (*CLOI*) is a repository for information, such as that listed above, that may be needed by one or more protocols. *CLOI* maintains this information through two structures, a neighbor table and a message pool, described in detail below, and it supports *services* that will fill these data structures either once or continuously, depending on the information.

CLOI was placed between the routing and MAC layers for two reasons. First, its location allows the interface to retrieve much of the information sent from the node onto the network as well as many incoming packets. The second reason is that it offers potential for abstraction of the link layer as suggested in [4]. The MAC and physical layers do not have a global vision of the network and cannot provide enough information about its state for automated use with *CLOI*.

Finally, *CLOI* has no authority to make any routing, node activation, or medium access decisions. *CLOI* simply acts as an interface to the protocols in the stack, allowing them to access common yet important information about the node and its neighbors that can be used to optimize the protocols' performance.

C. Information Sharing Structures

CLOI maintains both a neighbor table, storing information about the node and its neighbors, and a message pool, storing information about current packets waiting to be transmitted.

TABLE I
A NEIGHBOR TABLE IS KEPT AT EVERY NODE i WITH INFORMATION ABOUT ITSELF AND EACH OF ITS NEIGHBORS j.

ID	Locat.	ϵ_{rem}	Abilities	Entity	Congest.	LQ	Status
Id$_i$	x_i, y_i, z_i	ϵ_i	$V_{i,m}$, Conf$_{i,m}$	E_i	C_i	0	S_i
Id$_j$	x_j, y_j, z_j	ϵ_j	$V_{j,n}$, Conf$_{j,n}$	E_j	C_j	$LQ_{i,j}$	S_j
1	*20, 50, 4*	*0.5*	*Light, 0.7*	*Door*	0.2	0	On
2	*10, 100, 4*	*1*	*Temp, 1.0*	*N/A*	0.4	0.9	On

1) Neighbor Table: The neighbor table is comprised of the following fields, with an entry (row) for the node itself as well as an entry for each of the node's known neighbors.

- Node ID: this is a number or description that uniquely identifies a neighbor to the node. It can be either a locally unique ID or a global ID.
- Location: this specifies the geographical coordinates of the node.
- Remaining energy (ϵ_{rem}): this is a normalized measure of the node's remaining energy.
- Abilities: this specifies the node's sensing and compute capabilities, such as packet aggregation.
- Entity: some nodes may be attached to (or specifically monitoring) a specific target, such as sensors attached to a soldier. The entity field is an XML description of the target (or targets) being monitored by the node.
- Congestion: this is a normalized measure of the congestion at the node.
- SNR or P_e: this metric provides an evaluation of the quality of a link between the node and its neighbor.
- Status: this indicates whether the node is On or Off.

Not any service or protocol should be allowed to fill this neighbor table. For instance, the MAC layer should not assume that because it received a packet from neighbor *A*, node *A* will be On in the coming minutes of the runtime. Moreover, a node's abilities need not be updated periodically, but only when they change. Table I illustrates this structure.

2) Messages Pool: Others have proposed using a message pool that includes details about the received and sent messages [4]. We agree with the pertinence to use such a structure and propose incorporating the following fields:

- A unique packet identifier,
- An XML tag describing the data,
- The priority of the packet.

We summarize these elements in Table II. Such a structure, combined with the neighbor table, can help several layers make decisions about the routing or media access for a packet.

III. ADDITIONAL DESIGN ISSUES

A. Information Exchange and Frequency of Updates

Since it is obvious that the fields in the neighbor table must be kept up-to-date for maximum gain, *CLOI* needs a

TABLE II
A MESSAGE POOL IS KEPT AT EVERY NODE.

PacketID	Description	Priority
PID_1	XML tag	P_1
102	*Route Repair*	*0.9*

method of exchanging information with the *CLOI* of other nodes. This is accomplished by proactively sending out a *vector* of information. This vector includes some or all of the fields necessary to populate the neighbor table: *node ID, node location,* ϵ_{rem}*, abilities, entity, congestion, link quality* and *status*. These fields will automatically be filled by the *CLOI* of the sending node and read by the *CLOI* of the node's neighbors. In addition, specialized services and layers will have reading and writing access to the neighbor table through the *CLOI* interface. The information vector may be piggy-backed onto broadcast packets or sent as a special packet. The inherent mode of propagation for the information vector is one-hop broadcast.

Dedicated services within *CLOI* take care of updating the fields of the neighbor table and message lists. However, some protocols exchange important data about their status at pre-determined times (*e.g.*, GPSR [7]), and this may not agree with the schedule imposed by *CLOI* . Thus, *CLOI* has an automatic update knob that such protocols may control. At the protocol's request, *CLOI* will automatically perform an update function, sending out a vector with its information.

Other layers in the protocol stack may inform *CLOI* to update the neighbor table fields as well. For instance, the node activation layer should notify *CLOI* of the status of a neighbor when it receives explicit notification from that node. For example, in the case of PEAS node activation [2], a *probe reply* is a clear indication that the issuing node is active (on), and thus PEAS should notify *CLOI* to set the status of the issuing node to "on".

B. Extending the Architecture

The neighbor table and message pool should provide provisions to be extended as the need for additional fields arises by new or existing protocols. This will help to ensure compatibility with most designs. For the neighbor table, additional entries should be appended to the end of the table and added to the information vector. For example, additional fields could include a gradient in the case of GRAB [5], a trigger in $i3$ [6], an application cost in DAPR [3], etc. Additional fields in the message pool could be a description of the sensed data, *e.g.*, the value of the temperature within a region useful to a packet aggregation service located down the path to the sink.

C. Accessing the Structures

To guarantee the stability of the architecture, services and protocols should only be in contact with *CLOI* , and not the structures themselves. The *CLOI* interface provides input and output functions with a fixed syntax. To access a value for a field, *CLOI* has to be given the following parameters: node ID, an XML tag describing the field within the structure, and finally a value when writing data.

D. Important Services

Peripheral services need to be added to the protocol stack to sustain *CLOI* . These include the following:

- ID assignment service. This could be as simple as pre-determined ID numbers hard-wired in the sensor, or this could be a service-type identifier, also used for routing, as proposed in [8].
- Location service. The geographic coordinates of a sensor are indispensable and usually assumed in most research works in the field of sensor networks.
- Channel estimator. Several strategies are possible, from reading the back-off value for CSMA schemes, to one-hop packet delivery ratios, or measurements of the bit error rate.
- Remaining energy measurement. Although the battery voltage degrades in a non-linear fashion, this is a good indicator (when mapped through a look-up table) of the remaining energy at the node. A service is needed to access the current battery voltage and perform the mapping from voltage to remaining energy.

IV. Related Work

In [4], Culler et al. propose SNA, a new architecture whose goals are markedly different than ours. SNA tries to create an abstraction level above the MAC layer to compensate for the multiplicity of platforms available today. While different in their objectives, our two architectures seem compatible.

V. Conclusions and Future Work

We proposed an information-sharing architecture that facilitates horizontal and vertical cross-layer optimizations in wireless sensor networks. *CLOI* keeps updated information on the network state, the nodes' states and the messages to be sent. All layers have access to the information maintained by *CLOI* , which ensures that all layers of the protocol stack can benefit from cross-layer optimizations facilitated through information-sharing. Our future work will focus on including application requirement information to this general architecture.

References

[1] C. Merlin and W. Heinzelman, "Cross-Layer Gains for Sensor Networks", *Proc. DCOSS '06 Poster Session*, Jun. 2006.

[2] F. Ye, G. Zhong, J. Cheng, S. Lu, L. Zhang, "PEAS: A Robust Energy Conserving Protocol for Long-lived Sensor Networks", *ICDCS '03*, Rhode Island, 2003.

[3] M. Perillo and W. Heinzelman, "DAPR: A protocol for wireless sensor networks utilizing an application-based routing cost," *Proc. WCNC*, 2004.

[4] D. Culler, P. Dutta, C. T. Ee, R. Fonseca, J. Hui, P. Levis, J. Polastre, S. Shenker, I. Stoica, G. Tolle, and J. Zhao, "Towards a Sensor Network Architecture: Lowering the Waistline", *Proc. HotOS'05*, 2005.

[5] F. Ye, G. Zhong, S. Lu, L. Zhang, "GRAdient Broadcast: A Robust Data Delivery Protocol for Large Scale Sensor Networks", *ACM WINETS Journal, Vol. 11, No.2*, Mar. 2005.

[6] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, "Internet Indirection Infrastructure", *Proc. SIGCOMM'02*, 2002.

[7] B. Karp, H. T. Kung, "GPSR: Greedy Perimeter Stateless Routing for Wireless Networks", *Proc. MobiCom'00*, 2000.

[8] V. Lenders, M. May, and B. Plattner, "Towards a New Comunication Paradigm for Mobile Ad Hoc Networks", *Proc. .MASS'06, MHWMN'05 Workshop*, Oct. 2005.