Cross-Layer Protocols for WSNs: A Simple Design and Simulation Paradigm

Mohamed Hefeida*, Min Shen**, Ajay Kshemkalyani** and Ashfaq Khokhar*
* Department of Electrical and Computer Engineering
** Department of Computer Science
University of Illinois at Chicago, Chicago, IL 60607
Email: {mhefei2, mshen6, ajay, ashfaq}@uic.edu

***Abstract*—In this paper, we propose a Cross-Layer Application-aware Paradigm (CLAP) for designing and simulating Cross-Layer (CL) protocols. CLAP allows each layer to *publish* its local information to be shared with other layers and *subscribe* to other layers' shared information via an Information-Layer (I-Layer). The controlling layer, where optimization decisions are made, also utilizes the I-Layer to configure the behavior of other layers according to its current demands and their reported status. The publish/subscribe behavior is achieved by a new API designed as an augmentation to the SIDnet-SWANS simulator. This eliminates the need for bypassing/hacking conventional design hierarchies and simulator architectures, which greatly reduces the design and implementation complexities of CL protocols. CLAP facilitates CL interactions and extends the application layer's awareness and capabilities. This will lead CL protocol design in WSNs to a higher level of awareness via seamless CL information access and sharing, a new dimension of adaptability to operating conditions via continuous reconfiguration, and much simpler implementations.**

***Index Terms*—Cross layer design paradigms; design complexity; simulator hierarchies; wireless sensor networks**

I. INTRODUCTION

In wireless communication systems, the random time-varying nature of the channel leads to variations in system performance. Such variations can affect all conventional Open System Interconnection (OSI) layers and cause significant performance degradation. Although the OSI model standardizes, simplifies and accelerates efficient development of network protocols, it falls short in dealing with such variations. This is mainly due to the rigid boundaries set on the functionality of each protocol operating at each layer, and the limited information shared between adjacent layers.

In response to the above limitations, various studies explored possible gains from violating the OSI model when designing network protocols, which are referred to as Cross-Layer (CL) protocols. A wide variety of CL protocols exists in the literature, varying *from* utilizing CL information (information from a different layer) in optimizing the operation of another layer [1] *to* completely merging the functionality of several layers [2], [3]. Along with CL protocols, CL paradigms, aiming at standardizing the CL design process, also emerged [4]–[7]. Since most CL protocols do not follow a specific CL paradigm (e.g. [1], [8]), and many CL paradigms are tailored towards specific protocol optimizations (e.g. [9], [10]), in our brief review of those efforts, we refer to both as CL approaches.

CL approaches can be classified, based on their architecture and behavior, as being evolutionary or revolutionary [11]. Evolutionary approaches consider compatibility with the standard OSI model (i.e. incorporating CL capability into the OSI architecture), while revolutionary approaches only consider performance optimization and completely disregard the standard (i.e. several layers are melted into one). Examples of evolutionary CL approaches in Mobile Ad hoc Networks (MANETs) were presented in [4]–[7], [11], and a few studies adopting the revolutionary approach in Wireless Sensor Networks (WSNs) are discussed in [2]. A key difference between MANETs and WSNs, that must be considered in any CL approach for WSNs, is that the latter suffers much tighter energy constraints, due to infeasibility of battery recharging.

In this paper, we propose a Cross-Layer Application-aware Paradigm (CLAP), to facilitate CL protocol design in WSNs and overcome implementation complexities encountered in most CL approaches. CLAP incorporates an Information-Layer (I-Layer) as means of CL interaction. The I-Layer is accessible to all layers of the network stack through a *publish*-and-*subscribe* fashion. The I-Layer's architecture gives the application layer the capability of directly accessing lower layer(s) information and modifying their behavior(s), without following the conventional OSI hierarchy. This introduces a new level of application layer awareness and control over underlying protocols, which is a key distinction between CLAP and other CL approaches.

We implement an extension to the SIDnet-SWANS simulator [12] to incorporate CLAP. By designing the I-Layer as a hash table and providing an API for other layers to store and access information in it, we are able to realize the proposed paradigm in this simulator. Based on the simulator, we also demonstrate a sample scenario adopting the new paradigm. CLAP increases the level of adaption and awareness, while simplifying the design, implementation and operation of CL protocols. It allows changes made at the very top of the network stack to directly impact its very bottom, which gives CL protocol developers unprecedented design freedom and implementation simplicity.

The remainder of this paper is organized as follows: section II is a brief review of CL protocols and paradigms. Section

III discusses the details of CLAP, its architecture and implementation which extends the SIDnet-SWANS simulator. An example CL protocol, utilizing CLAP, is presented in section IV, followed by the conclusion and future work in section V.

II. Related Work

It is suggested by many studies that the revolutionary CL approach is more suitable for emerging technologies, such as WSNs [11], [13]. This can be true when the application operates in a standalone fashion or requires minimal compatibility. However, many WSN applications are integrated and interfaced with larger, conventionally designed networks (e.g. [14]). For such applications, sacrificing interoperability and compatibility, as suggested in revolutionary CL approaches, must be reconsidered. We use the terms conventionally and OSI-Like interchangeably.

Realizing limitations due to absence of modularity in revolutionary CL approaches, many research efforts adopted the evolutionary approach in proposing CL paradigms. In [6], CL information reflects a layer's state and can only be communicated between adjacent layers. Attempting to regulate CL interactions, information from non-adjacent layers is accessed via a mapping of that information to the adjacent layer. In [7], CL information is stored and organized via a network status entity that preserves layer separation. Finally, in [5], a similar architecture to that in [7] is proposed, however global network information is piggybacked over each data packet and factored in the optimization decision. All the above CL paradigms mainly target MANET environments with less critical energy constraints, compared to WSNs, and do not give in-depth details about an actual protocol operation. Moreover, they treat the application layer as a client that demands service without being involved in operation of lower layers. A comparison between several CL paradigms is presented in [5].

In addition to the above studies proposing new CL paradigms, other efforts developing CL protocols, not adhering to any CL paradigm, also exist. These efforts vary *from* utilizing a single parameter from one layer in the functionality of another *to* merging the functionality of two layers [13], [15]. For example, in [1], next-hop information from the routing layer is utilized at the MAC layer to schedule multi-hop traffic flows. The shared information was extended in [8], where all packets in the routing buffer were considered in the scheduling process and led to significant performance gains. On the other hand, in [3], a joint scheduling and routing scheme was proposed (i.e. completely melting routing and MAC). The main differences between CL protocols are (i) the type and amount of CL information communicated/shared and involved in the design, and (ii) which layers interact and how.

Although CL information utilized is different and for different purposes, the evolutionary CL protocols discussed above, and many others summarized in [13], [15], share many of the following design and implementation aspects and limitations:

- expose CL information, that is hidden or considered irrelevant in conventional designs, which led to significant performance improvements.
- CL interaction is usually tailored to one layer and a few parameters (e.g. packet similarity/order in routing buffer in [8])
- application layer interaction/involvement in lower layer operation is very limited (e.g. overheard packets in [16]).
- were implemented via CL operations on simulators designed with OSI-like architectures at the core of their operation, which influenced the CL design and significantly increased implementation complexity.

The design complexity of CL protocols is partly due to incorporating CL information in the design process; however, a greater complexity lies in the implementation and instability of such protocols. The implementation complexity results from the need to *bypass/hack* simulator hierarchies that originally targeted non-CL simulations. This strongly contradicts the main purpose of discrete-event simulations which aim at reducing complexity [17]. This is also a main reason for the limited presence of CL paradigms in the design process of various CL protocols (i.e. limited publicity of CL paradigms beyond the proposing research group).The instability encountered in any CL approach results from often foreseen joint optimization of layers. That is, feedback from one layer affects the operation of another, which creates a closed-loop feedback system [18].

III. CLAP Design

In this section, we illustrate CLAP's design by comparing it to conventional paradigms and emphasize its novelty compared to state of the art in CL design. We also cover CLAP's implementation details which extends SIDnet-SWANS [12].

A. Comparing to Conventional Designs

We present a new CL paradigm to facilitate the design of CL protocols for WSNs in which CL interaction and control is achieved via the I-Layer (Information-Layer). The I-Layer is accessible to all other layers of the network stack and contains all information shared between them. The information stored in the I-Layer can either be status or control information. One of the layers (the application layer in the proposed scenario) is granted control over CL operation and hence dictates how the information is utilized (e.g. tradeoff energy consumption for throughput). The controlling layer publishes control information, reflecting the desired behavior, to which other (non-controlling) layers subscribe and adapt their behavior accordingly. Non-controlling layers publish status information, to which the controlling layer subscribes, as discussed in section III-B.

Figure 1 illustrates the main differences between conventional (OSI-like) CL information sharing/interaction and CLAP. For presentation simplicity, we integrate the transport layer into the routing layer and the physical layer into the MAC layer. In CLAP, assuming application control of CL interaction as in Fig. 1(b), I_A captures the application decision to modify the underlying MAC and/or routing operation. However, in the conventional case, I_A only represents CL info (i.e.

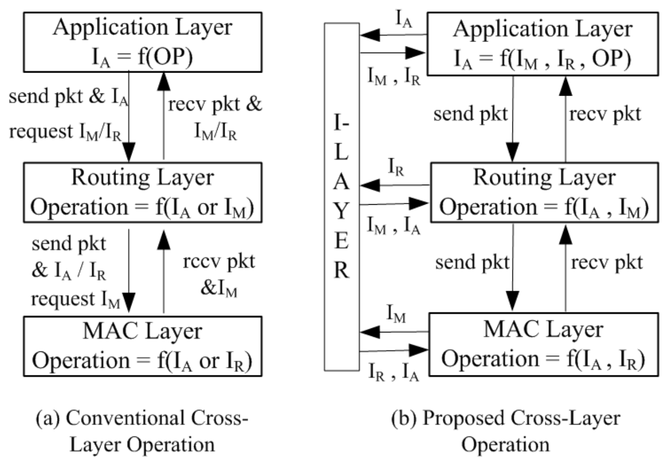
Application Layer
$I_A = f(OP)$
send pkt & I_A request I_M/I_R
recv pkt & I_M/I_R
Routing Layer
Operation = $f(I_A$ or $I_M)$
send pkt & I_A / I_R request I_M
recv pkt &I_M
MAC Layer
Operation = $f(I_A$ or $I_R)$
(a) Conventional Cross-Layer Operation
I-LAYER
I_A
I_M , I_R
Application Layer
$I_A = f(I_M , I_R , OP)$
send pkt
recv pkt
I_R
I_M , I_A
Routing Layer
Operation = $f(I_A , I_M)$
I_M
I_R , I_A
MAC Layer
Operation = $f(I_A , I_R)$
(b) Proposed Cross-Layer Operation

Fig. 1. A comparison between CL protocol operation in (a) conventional paradigm and (b) CLAP: I_A, I_R and I_M denote information shared by Application, Routing and MAC layers, respectively. The functionality of each layer is denoted by f( ), and OP represents Optimization Parameters.

no control info). In CLAP, CL information is communicated and handled via the I-layer, which allows non-adjacent layers to interact without *unnecessary* involvement of other layers. For example, the application and MAC layer can interact without routing layer involvement. This would greatly simplify the design and implementation of protocols involving such CL interaction [16]. In [16], the overheard data packets (at the MAC layer) are processed at the application layer which suppresses packets to be sent, if it detects correlation between them and the overheard packet. In addition, accessing other layers' information does not require permission nor hacking the design hierarchy, it is accessible via the I-Layer. Another advantage of CLAP is the ability of involving more than two layers in the interaction and no restriction on what information can be shared. This makes tri-layer interaction possible as discussed in section IV.

Although the idea of providing means of CL interaction, involving more than two layers, is not new [7], the application's involvement/control capabilities over the protocol as well as the information sharing technique are unique to CLAP. Moreover, studies like [7] mainly target MANETs, where energy constraints are considerably looser (i.e. no duty cycling), compared to WSNs, and hence, suffer less performance instability [19].

To demonstrate the unique capabilities of CLAP, we consider the case of application control over protocol behavior. Such control is embedded in CL control information (I_A in Fig. 1(b)) and its details are decided by the protocol designer (see section IV for an example). This gives the application knowledge about underlying layers' operating conditions and direct control over their operation. For example, in a conventional scenario, the running application is not aware and has no control over the transmission *retry limit* which is solely set by the underlying MAC scheme (retry limit defined and illustrated in section IV). In CLAP, the application layer would not only be aware of the retry limit set by the MAC protocol, it would have the ability to change it according to its observations (I_M and I_R) and its needs (reflected in OP), resulting in an application-controlled customizable and reconfigurable MAC operation. More details about application awareness, optimization parameter formulation and interaction with other layers can be found in [10].

Notice that all protocols in the literature, where CL interaction is limited to information sharing (e.g. MAC and routing interaction in [8]), can still be implemented using CLAP. In such scenario, the controlling and controlled layer are the same (i.e. no control info will be published). However, more efficient and seamless access to CL information can be achieved via the I-Layer (i.e. no need for extra control messages to request access to CL info and no complex hacking for simulator hierarchy). Moreover, CLAP can implement non-CL protocols, which guarantees compatibility, however at the cost of the non-utilized CL capability overhead (i.e. I-Layer).

B. *CLAP Implementation*

CLAP's novelty lies in the following unique features:

- CL information is encapsulated in the I-Layer via a *publish*-and-*subscribe* manner, which eases cross-layer information sharing by enabling direct interaction between any two layers without having to go through others.
- allowing all layers *(including the application layer)* to publish and subscribe to the I-Layer, which makes complex CL interactions, especially those involving more than two layers, easier to control, manage, and implement.

1) Architecture: The implementation is done on SIDNet-SWANS. Fig. 2 depicts the architecture of the system and our extensions are highlighted. SWANS provides the implementation to represent the network stack of each node in the simulation, while SIDNet provides a user-friendly GUI together with a NodeAPI, giving the user run-time information, to better manage all the nodes [12]. Our implementation utilizes the advantages of SIDNet-SWANS, and supports CL interaction by extending the behavior of the node stack and NodeAPI, in order to integrate the I-Layer into the system. Layers that are involved in CL interactions will publish their CL information into the I-Layer and subscribe to CL information of interest that is published by other layers. The I-Layer itself acts as storage for CL information and demands. This is

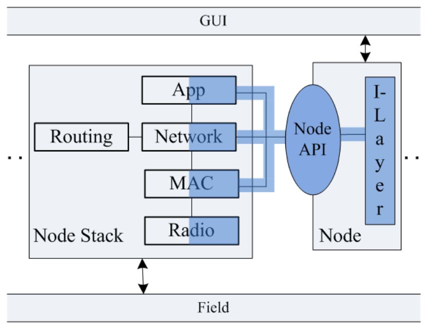

Fig. 2. System Architecture: Modifying CL interaction in the node stack by enabling CL information sharing via I-Layer through an extended NodeAPI interface. Modifications/extensions are highlighted.

achieved via the publish and subscribe actions of other layers. In the remaining part of this section, we will discuss those functionalities in detail.

2) I-Layer Hookup: Being a layer that is not part of the traditional network stack, the hookup of the I-Layer can be non-trivial. However, the implementation of SIDNet-SWANS makes this task relatively simple and straightforward. From Fig. 2 we can see that, for each node, all the layers involved in CL interactions are associated within the *Node* object. In the implementation of SIDNet-SWANS [12], the class defining each layer includes the Node object as an instance variable. This makes the Node class a reasonable choice for placing the I-Layer, since all the layers will have an easy access to the I-Layer through the Node object. In our implementation, we assign an *I-Layer* object as an instance variable in the Node class. We also extend the methods in the *NodeAPI*, and the interface of the Node class to expose methods interacting with the I-Layer. Hence, through the NodeAPI, all the layers involved in the CL interaction will have a unified access to the I-Layer. The details of how this access occurs is discussed in sections IV-B and IV-C.

3) Publishing CL Information: Before publishing CL information into the I-Layer, each type of information has to be assigned a key which has to be agreed on among all layers taking part in the CL interaction. When some layer *l* wants to publish CL information with key *k*, it calls the *publish(key)* method in the NodeAPI as shown in Fig. 3. This will trigger the control unit of I-Layer to put a new key-value pair with key *k* in the internal hash table of the I-Layer. The control unit will also remember that the entry with key *k* in the hash table can only be updated by layer *l*. When layer *l* has the actual data to write into I-Layer, it will call the *update(key, value)* method in NodeAPI. This will trigger the control unit to verify the caller being layer *l* and update the entry with key *k* in the hash table.

4) Subscribing to CL Information: CL information sharing not only requires publishing information to the I-Layer, but also subscribing to the I-Layer in order to access other layers' shared information. The subscribing part of the interaction works as follows. Since the layers have agreed upon the keys for all CL information, if layer *l* wants to subscribe to a particular type of information with key *k*, it only needs to call the *subscribe(key)* method in NodeAPI (Fig. 3). However, before that, it also needs to call the *getKeyList( )* method

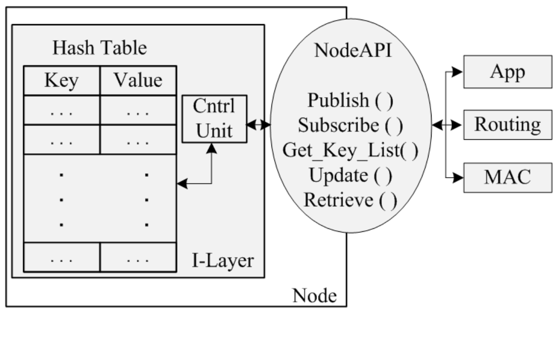

Fig. 3. CL information shared in a publish-and-subscribe manner.

in NodeAPI to verify that the information has already been published. *getKeyList( )* tells the control unit of the I-Layer to return all the keys in the hash table as a list. The caller, layer *l*, will then be able to see whether *k* is within this list. If the verification succeeds, layer *l* can then call the *subscribe(key)* method. This will cause the control unit to remember that the entry with key *k* in the hash table will be read by layer *l*. After calling *subscribe(key)*, layer *l* will periodically check the I-Layer by calling *retrieve(key)* to gain the current value of the CL information it is interested in. Calling *retrieve(key)* in *NodeAPI* will tell the control unit of the I-Layer to check if layer *l* is among those layers that are interested in information associated with key *k*, and return the current value of the entry with key *k* in the hash table. Our implementation also supports subscribing to the CL information dynamically. If some layer wants to start subscribing to a new type of information at run time, it just needs to call *getKeyList()* first to verify the desired information is already published. The control unit of the I-Layer will cover the rest as described above.

IV. Sample Scenario

A. Overview

In this section, we introduce a sample scenario developed based on CLAP. As mentioned in section III-A, we consider the case of application control over CL interaction and protocol operation. The application layer monitors status info, published by MAC and routing layers, by subscribing to the I-Layer. The user is provided with an interface to both observe and specify values (i.e. run-time application preferences) of the status information currently being monitored [10]. Upon receiving user input, the application layer processes user demands, based on which it issues control instructions to the underlying layers by publishing the instructions into the I-Layer. Underlying layers, which subscribe to control instructions published by the application layer, will take corresponding actions in an attempt to draw current status towards the user's/application's desired status. The details of this process are explained in the following subsections.

B. Monitoring and Control

In CLAP, the application's decision to modify the underlying protocol's behavior is based on the current status information, both local to the application layer (e.g. remaining battery) and that reported by other layers, and the run-time application/user preferences represented by target values for specific status information (e.g. delay, power consumption etc.). We define the set of target values $T^{'} = T^{'}_1, T^{'}_2, ..., T^{'}_i$. Each element of $T^{'}$ is assigned a weight W_i based on its priority among the other target elements, where $\Sigma_i W_i = 1$, and has a current value T_i and $T_i \in T$, where T is the set of current values of all elements in $T^{'}$. The distance between T_i and $T^{'}_i$ is D_i which is computed as $D_i = W_i \frac{|T_i - T^{'}_i|}{T^{'}_i}$. Both W_i and $T^{'}_i$ are assigned by the application along with an error tolerance, tol_i. These assignments can be changed (reconfigured) according to the application demands and observed network operating conditions. We define a set of possible actions, A, the application can execute to modify underlying MAC and/or routing functionality where $A = \{A_1, A_2, ..., A_i\}$. Note that for presentation simplicity, we assume one action per status reported (T_i), however, each T_i can have several actions (A_is) affecting it, as shown in Table I.

Without loss of generality, we assume the application is aware of the following:

- current average power consumption (T_1)
- current average delay (T_2)
- current average delivery ratio (T_3)
- desired target values of the above parameters (i.e. $\{T_1^{'}, T_2^{'}, T_3^{'}\}$) and their weights (i.e. $\{W_1, W_2, W_3\}$)

The application can modify routing and MAC operation via the following set of actions:

- retry limit (A_1)
- duty cycle (A_2)
- order packets in routing buffer based on destination (A_3)

Note that the above awareness info/actions can be extended or limited by the protocol designer.

We define delay as the time a packet spends in the routing buffer; delivery ratio is the ratio of number of packets successfully transmitted to the total number of packets sent. All monitored parameters (status info) are averaged over the same time interval. Retry limit is the number of attempts by MAC layer to send a packet before dropping it, and duty cycle is the ratio between active period duration and total frame duration (i.e. *active* + *sleep*).

C. Processing User Input

In the scope of the capabilities and awareness described in section IV-B, we explain the application-layer decision making process. The application is capable of increasing/decreasing the retry limit and/or duty cycle and/or reordering packets in the routing buffer. The first two capabilities/actions have a direct effect on power consumption and delay; however, reordering packets does not. Increasing the retry limit is expected to increase power consumption and delay, however, it is also expected to increase delivery ratio, and vice versa. Increasing the duty cycle is expected to reduce delay and increase power consumption, and vice versa.

The relationship between the above status info (T) , user/application demands (T'), and possible actions (A), is greatly dependent on operating conditions (e.g. congestion, network topology, traffic patterns etc.). This makes the quantification of such relationships infeasible. Therefore, the application adopts an exhaustive search for the combination of actions that will steer the functionality of underlying layers in order to meet its own demands (i.e. target values). However, there are no guarantees that the application demands will be achieved, and in such case the initial state (prior to applying the search) is retrieved. The general relationship between application demands and related actions is illustrated in Table I. Note that any combination of such demands is possible. Although any increase in retry limit is expected to increase duty cycle, we assume that the increase in retry count encountered by one packet is offset by a decrease in that of another packet sent in the same cycle, and hence consider both actions to be independent. This assumption is based on the MAC protocol capability of sending multiple packets/frame.

TABLE I
DEMAND/ACTION RELATIONSHIP

Application demand	Related action
$\pm delivery\ ratio$	$\pm retry\ limit$
$\pm delay$	$\pm retry\ limit$; $\mp duty\ cycle$
$\pm power\ consumption$	$\pm retry\ limit$; $\pm duty\ cycle$

D. Simulation Results

We use CLAP to implement the sample scenario explained in section IV-C. The underlying MAC protocol is SMAC [20] and we assume that single schedule and routing information are available to nodes as in [1], [8]. We simulate two networks, a 17-node cross-chain, and a 25-node grid. Each simulation lasts for 2000 seconds. On both networks, the distance between adjacent nodes is 200 meters. The simulation parameters are similar to those in [1], [8].

The main goal of our simulations is to demonstrate CLAP's ability of tailoring main performance metrics, namely, delivery ratio and delay. Energy consumption is kept constant by keeping a constant duty cycle throughout the simulations. Delay and delivery ratio were defined in section IV-B.

Fig. 4 shows how increasing the retry limit can increase delivery ratio. When the data rate is at 1 packet/second and the retry limit is increased from 2 to 7, delivery ratio is increased from 84.6% to 90%. This modest 6.4% improvement is offset by a 7.8% increase in delay as shown in Fig. 5. This delay penalty also offsets the delivery ratio gains at the lower data rates. This is due to the bottle neck occurring at the center

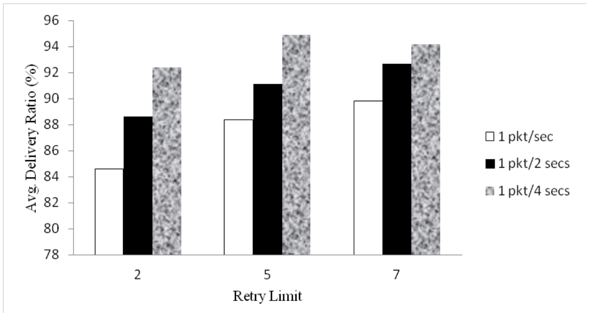

Fig. 4. Average delivery ratio of the 17-node cross-chain network at various data rates and retry limits.

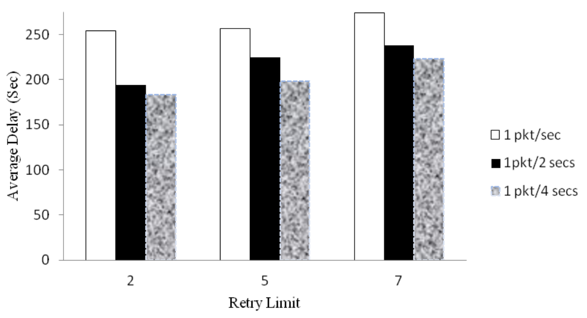

Fig. 5. Average delay of the 17-node cross-chain network at various data rates and retry limits.

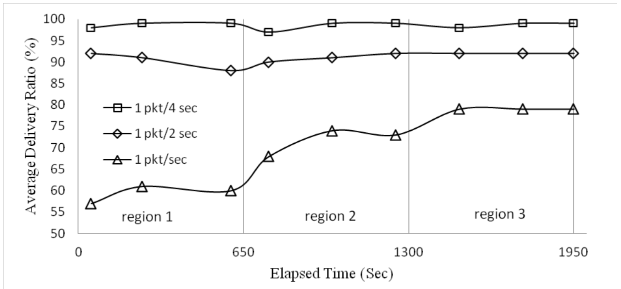

Fig. 6. Average delivery ratio of the 25-node grid. Retry limit increased from 2 in region 1 to 5 in region 2, then to 7 in region 3.

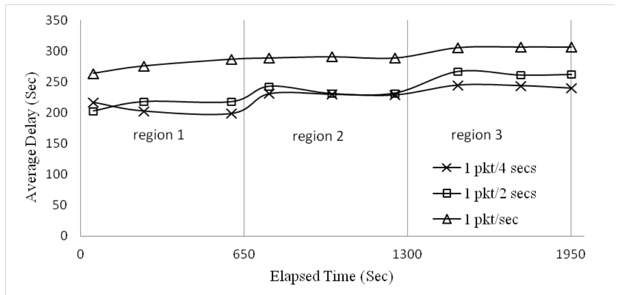

Fig. 7. Average delay of the 25-node grid. Retry limit increased from 2 in region 1 to 5 in region 2, then to 7 in region 3.

node (at the crossing of the two chains) which greatly limits the traffic flow of both routes. Note that cross-chain results were averaged over the entire simulation duration and over the entire network.

Fig. 6 shows the effect of CLAP's response to the user's (application's) demand of increasing delivery ratio by increasing the retry limit. The delivery ratio increases from 57% to 79% when the data rate is at 1 packet/second. This 38.6% increase in delivery ratio causes an increase in delay of only 16.8%, as shown in Fig. 7. The least improvement in delivery ratio occurs at the lowest data rate (1 packet/ 4 seconds). This is due to the very small margin available for any improvement in delivery ratio, as it is already at 98% when the retry limit is 2. This is reflected in the delay results at lower data rates, as shown in Fig. 7.

V. Conclusion and Future Work

The complexity of developing CL protocols in WSNs should not be affected by lack of CL information that is hidden in conventional single layered design approaches and their complementing simulators. Such conventional hierarchies influence the design process and add significant complexity to CL protocol implementation. Towards facilitating CL protocol design and simulation, we presented a new Cross-Layer Application-aware design Paradigm (CLAP). CLAP incorporates an I-Layer to efficiently moderate CL interactions. Any layer can access the I-Layer for monitoring/updating/control purposes. Not only this allows any combination of CL interactions to be realized by CLAP, it also grants the application layer new awareness and control capabilities. We integrated CLAP into SIDnet-SWANS [12] and simulated a sample scenario. The simulated scenario allows the user (application) to steer the operation of the underlying protocol(s), resulting in a reconfigurable CL application-controlled protocol.

In our current setup, the user is assumed to be configuring the application needs. This assumption limits the application scope of the proposed design approach. To expand the applicability of CLAP, we plan to extend its functionality to allow remote reconfiguration.

References

[1] S. Du, A. Saha, and D. Johnson, "RMAC: A routing-enhanced duty-cycle MAC protocol for wireless sensor networks," in *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, pp. 1478–1486, May 2007.

[2] M. Vuran and I. Akyildiz, "Xlp: A cross-layer protocol for efficient communication in wireless sensor networks," *IEEE Transactions on Mobile Computing*, pp. 1578–1591, 2010.

[3] S. Cui, R. Madan, A. Goldsmith, and S. Lall, "Joint routing, mac, and link layer optimization in sensor networks with energy constraints," in *Communications, 2005. ICC 2005. 2005 IEEE International Conference on*, vol. 2, pp. 725–729, IEEE, 2005.

[4] V. Srivastava and M. Motani, "Cross-layer design: a survey and the road ahead," *Communications Magazine, IEEE*, vol. 43, no. 12, pp. 112–119, 2005.

[5] R. Winter, J. Schiller, N. Nikaein, and C. Bonnet, "Crosstalk: Cross-layer decision support based on global knowledge," *Communications Magazine, IEEE*, vol. 44, no. 1, pp. 93–99, 2006.

[6] R. Knopp, N. Nikaein, C. Bonnet, H. Aiache, V. Conan, S. Masson, G. Guibe, and C. Martret, "Overview of the widens architecture, a wireless ad hoc network for public safety," in *IEEE SECON 2004*, 2004.

[7] M. Conti, G. Maselli, G. Turi, and S. Giordano, "Cross-layering in mobile ad hoc network design," *Computer*, vol. 37, no. 2, pp. 48–51, 2004.

[8] T. Canli, M. Hefeida, and A. Khokhar, "BulkMAC: A cross-layer based MAC protocol for wireless sensor networks," in *Proceedings of the International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 442–446, 2010.

[9] A. Safwat, "A novel framework for cross-layer design in wireless ad hoc and sensor networks," in *Global Telecommunications Conference Workshops, 2004. GlobeCom Workshops 2004. IEEE*, pp. 130–135, IEEE, 2004.

[10] M. Hefeida, T. Canli, A. Kshemkalyani, and A. Khokhar, "Context modeling in collaborative sensor network applications," in *Collaboration Technologies and Systems (CTS), 2011 International Conference on*, pp. 274–279, IEEE.

[11] F. Aune, "Cross-layer design tutorial," *Norwegian University of Science and Technology, Dept. of Electronics and Telecommunications*, 2004.

[12] O. Ghica, G. Trajcevski, P. Scheuermann, Z. Bischof, and N. Valtchanov, "Sidnet-swans: a simulator and integrated development platform for sensor networks applications," in *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pp. 385–386, ACM, 2008.

[13] T. Melodia, M. Vuran, and D. Pompili, "The state of the art in cross-layer design for wireless sensor networks," *Wireless Systems and Network Architectures in Next Generation Internet*, pp. 78–92, 2006.

[14] R. Bajwa, R. Rajagopal, P. Varaiya, and R. Kavaler, "In-pavement wireless sensor network for vehicle classification," in *Proceedings of the IEEE International Conference on Information Processing in Sensor Networks (IPSN), 2011*, pp. 85–96.

[15] B. Yahya and J. Ben-Othman, "Towards a classification of energy aware mac protocols for wireless sensor networks," *Wireless Communications and Mobile Computing*, vol. 9, no. 12, pp. 1572 – 1607, 2009.

[16] Y. Iima, A. Kanzaki, T. Hara, and S. Nishio, "Overhearing-based data transmission reduction for periodical data gathering in wireless sensor networks (pdf)," 2009.

[17] J. Henriksen, "Taming the complexity dragon," *Journal of Simulation*, vol. 2, no. 1, pp. 3–17, 2008.

[18] V. Kawadia and P. Kumar, "A cautionary perspective on cross-layer design," *Wireless Communications, IEEE*, vol. 12, no. 1, pp. 3–11, 2005.

[19] J. Garcia-Macias and J. Gomez, "Manet versus wsn," *Sensor Networks and Configuration*, pp. 369–388, 2007.

[20] W. Ye, J. Heidemann, and D. Estrin, "An energy-efficient mac protocol for wireless sensor networks," in *Proceedings of the IEEE INFOCOM*, vol. 3, pp. 1567–1576, 2002.