X-Layer: An Experimental Implementation of a Cross-Layer Network Protocol Stack for Wireless Sensor Networks

Aaron Beach, Mike Gartrell, Saroch Panichsakul,
Li Chen, Chao-Kai Ching, and Richard Han
Contact: Aaron.Beach@colorado.edu, Mike.Gartrell@colorado.edu

Department of Computer Science
University of Colorado at Boulder

Technical Report CU-CS-1051-08

December 2008

X-Layer: An Experimental Implementation of a Cross-Layer Network Protocol Stack for Wireless Sensor Networks

Aaron Beach, Mike Gartrell, Saroch Panichsakul, Li Chen, Chao-Kai Ching, Richard Han
Department of Computer Science, University of Colorado at Boulder
Contact: Aaron.Beach@colorado.edu, Mike.Gartrell@colorado.edu

Abstract

Cross-layer design of network protocol stacks is important for wireless networking applications due to limited resources. Although much has been proposed in the literature, little experimental cross-layer work has been done. Even less has been done concerning actual designs for such cross-layer network stacks. This paper presents X-Layer, an implementation of a "cross-layer" network stack for wireless sensor networks, designed under the assumption that different network layers will interact and share resources directly with each other. X-Layer's cross-layer stack is compared with a network stack implemented with traditional network layer abstractions using the same basic protocols at each layer as in the cross-layer stack. The efficiency and effectiveness of the stacks are compared and evaluated. Based on this evaluation a new mechanism is proposed for managing shared network resources and code in an orderly fashion.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication, Distributed networks, Network communications, Network topology*

General Terms

Performance, Design, Experimentation

Keywords

Cross-Layer Design

1 Introduction

Cross-layer approaches towards network protocol design have held much promise in recent years [1, 2, 3, 4]. Indeed, our contention is that cross-layer design of network protocol stacks makes its most compelling case for improving network performance in the field of wireless sensor networks (WSNs), due to the extreme resource constraints posed by the sensor node devices and by their relatively low bandwidth RF radios. However, most current proposed approaches towards cross-layer design have not empirically validated their theoretical ideas with a detailed implementation and evaluation on a real world system. In this paper, we present X-Layer, which we believe is the first experimental implementation and evaluation of a cross-layer network protocol stack in WSNs on standard WSN hardware and software.

Cross-layer design contends that increasing direct interactions between each of the layers in a network protocol stack leads to improved performance and lower resource usage, at some cost in complexity of debugging. Traditional network protocol stacks have been designed with independent layering in mind, e.g. the Internet protocol stack, such that each layer is only able to communicate with the layer above it and below in the stack. This independently layered approach led to great leaps forward in innovation once the service interface between each layer was standardized, i.e. as long as network designers knew what services were expected of each layer, then they could go off and introduce the necessary protocol functionality underneath these service interfaces and independently optimize each layer.

While this independently layered approach towards network protocol stack design has paid great dividends in the IP world, its extension to the world of WSNs is more problematic, primarily due to the resource limitations of sensor nodes. Independent layering comes at a cost in system performance and resources. First, independent layering is suboptimal if a layer L only knows about its neighboring layers L-1 and L+1. In contrast, if a layer L can base its adaptation decisions by accessing all the performance parameters of all N layers of a network stack, then it can make a more informed and hence more efficient adaptation decision. Second, independently designed layers often replicate buffers, variables, data structures, e.g. neighbor tables, and functionality across layers, e.g. link state estimation, with only limited regard as to resource/memory consumption. This is because most network protocols operate on sufficiently resource-rich platforms, e.g. PCs and laptops with several GB of RAM, where designers don't have to be overzealous about minimizing resource consumption. Even many of today's cellular smartphones have hundreds of MB of RAM. One of the smallest TCP/IP stacks known consumes at least 5 KB of RAM [5]. Even this is too large for many of today's

standard WSN nodes, which may allocate at most 1 KB of RAM to the entire network stack.

Thus, we feel that sensor networks present the strongest case yet made for cross-layer design of the network protocol stack. Current implementations of network stacks in sensor networks follow a more traditional independent layering approach [6, 7]. The alternative architecture of a cross-layer network protocol stack is shown in Figure 1. We observe that there are ${}_NC_2$ pairwise combinations, including both the traditional neighbor-layer interactions of an independently layered network stack (arrows down the center) as well as "bonus" pairwise interactions between non-neighboring layers (arrows on either side of the stack). These bonus interactions represent opportunities for the cross-layer approach to improve performance over the traditional independently layered network stack. Also, though not shown, data structures and buffers can be shared across many layers, reducing memory consumption.

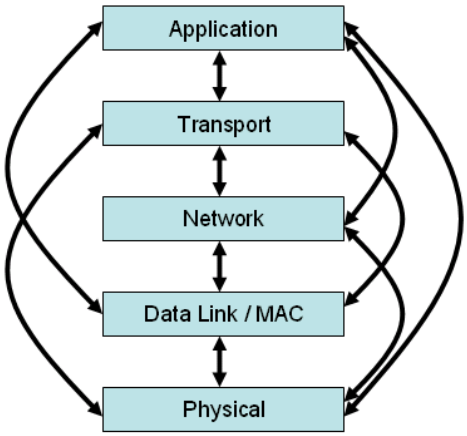

Figure 1. X-Layer's cross-layer network protocol stack.

The figure raises several intriguing broader research questions. Which sets of pairwise cross-layer interactions yield the greatest performance improvement in the network? More specifically, what data variables and structures at what layers are most pivotal to achieving improved performance? Given the possibility of feedback loops, what can be done to mitigate instability? How can such a cross-layer network stack be most efficiently and effectively realized? All of these research questions certainly deserve the WSN research community's attention. However, the first and likely most important research question, which is answered by this paper, is whether we can show in a real world system that there is any measurable benefit at all to network performance due to cross-layer interactions, i.e. can we find at least one subset of pairwise cross-layer interactions that demonstrably improves the network? Our research community requires an empirical proof as to the efficacy of cross-layer approaches.

The main contribution of this paper is that it *quantifies* the real-world benefits of cross-layer design by building a complete cross-layer network stack solution on standard WSN hardware and software. We choose a select subset of pairwise interactions, and validate that this subset achieves superior joint performance when compared to a more traditional independently layered stack, as measured in network throughput, latency, and energy lifetime. The challenges that we have overcome to accomplish this research include implementing each of the layers of our network protocol stack in both their independently layered forms and their cross-layered variants, selecting the specific cross-layer interactions to study, and designing and executing the experiments to evaluate these differing approaches in a testbed environment.

In the following, Section 2 summarizes the related work in cross-layer design. Section 3 describes the design and implementation of the X-Layer cross-layer network stack. Section 4 explains the evaluation of X-Layer and the experimental results, including several observations that lead to a comprehensive new architecture for realizing cross-layer network stacks. Section 6 describes the rich future work left to be explored in this promising WSN research area. Section 7 concludes the paper.

2 Related Work

We classify prior work in cross-layer network research into three categories, namely research into cross-layer WSNs, mobile ad hoc networks (MANETs), and wired TCP/IP networks.

2.1 Wireless Sensor Networks

Almost all prior work regarding cross-layer networking for WSNs has involved algorithmic analysis and/or simulation work, rather than building and evaluating an implementation of a cross-layer network stack. Since energy is a limited resource on wireless sensor nodes, much of the prior work has focused on energy conservation. [8] reviews approaches for optimizing at each layer of the WSN network stack and discusses motivations for cross-layer design, but does not provide a formal analysis of results. The main focus in this paper is on theory and design. [9] presents a cross-layer stack for WSNs that uses interaction between the MAC protocol and the routing protocol to minimize energy consumption; the primary focus here is on theory and simulation work. van Hoesel et. al. compare their cross-layer approach to traditional SMAC and DSR protocols, and show using simulation results that their cross-layer protocols achieve longer network lifetime. In a paper focused on simulation results, Sichitiu [10] describes a cross-layer interaction between the routing and MAC layers to conserve energy by setting up a synchronized off-on schedule for the wireless sensor nodes in a WSN. Simulation results verify that this energy conservation scheme can significantly increase network lifetime for certain types of networks. With a basis in theory and simulation, [11] proposes two energy efficient schemes for wireless ad-hoc and sensor networks that uses cross-layer interactions between the routing layer and MAC layer. In the first scheme, called Energy-Constrained Path Selection (ECPS), interactions between the MAC and routing layers are used to choose a route that maximizes the probability that the packet will reach the destination in at most some number of n transmissions. The second scheme,

called Energy-Efficient Load Assignment (E2LA), uses interactions between the MAC and routing layers to efficiently distribute load among a set of routes according to successful packet transmission probability. The successful packet transmission probability used in this paper is similar to the Expected Number of Transmissions (ETX) metric used in our routing layer.

In [12], an algorithm for a cross-layer joint optimization problem is presented that seeks to maximize WSN lifetime. Joint optimal design of the physical, MAC, and routing layers is considered in this algorithm, and the authors perform numerical studies to show the benefits of their approach. [13] uses numerical analysis to survey energy consumption in WSNs using cross-layer models that jointly consider topology, the MAC layer, and radio transceiver energy consumption. The authors reach several interesting conclusions. One such conclusion is that with a realistic radio model and certain network conditions, single-hop communication can be more energy efficient than multi-hop communication. Both [12] and [13] focus primarily on theory and numerical studies and analysis.

Cross-layer interactions in network stacks for WSNs impact the architecture and design of such stacks. Although Chameleon [6] is not a cross-layer network stack, the Chameleon architecture for sensor networks uses cross-layer bit-packed header fields to reduce the size of packet headers and conserve memory. Chameleon also implements cross-layer information sharing. SP [14] is also not a cross-layer network stack, but it does enable some cooperation between the network and link layers by implementing a shared neighbor table and message pool.

2.2 Mobile Ad-hoc Wireless Networks

There has also been much research into cross-layer networking for Mobile Ad hoc Wireless Networks (MANETs), since it is believed that a strictly layered approach inhibits the performance optimizations needed to deal with the dynamic mobile and energy-constrained environments [15, 16]. Most of this work involves theoretical and/or simulation-based work, rather than evaluation of a cross-layer network stack implementation for MANETs. [15] proposes the MobileMan architecture for MANETs, which allows different layers to cooperate by sharing network status information. No evaluation of results is presented in this paper. [16] makes the case for the importance of cross-layer design in ad hoc wireless networks and reviews opportunities for beneficial cross-layer interactions between different layers in the network stack. Both [15] and [16] focus on theory and design and do not present simulation or implementation results.

In [17], Kawadia and Kumar present simulation results for several cross-layer interactions in a wireless ad hoc network. Their simulation results for examples involving an adaptive rate MAC protocol and adaptation of transmit power show that some cross-layer interactions can lead to unstable loops and an unintended negative impact on system performance. In [18] the authors discuss simulation results for a wireless ad hoc network that show improved application performance resulting from cross-layer interactions between the physical, data link, and network layers.

While all of the papers mentioned so far in this MANET subsection focus on theory, design, and/or simulation, [19] describes an actual implementation of a cross-layer interface for interaction between layers, called Xian. Xian is intended to facilitate cross-layer experiments on MANET testbeds, and is implemented as a set of Linux kernel space and user space components. Xian enables interaction between adjacent and non-adjacent layers. In the examples discussed in this paper, upper layers are able to access metrics provided by the 802.11 MAC layer. While the architecture of this system is interesting, clearly this particular implementation is not appropriate for resource-constrained wireless sensor nodes.

2.3 Wired Internet

Since it can be argued that the success of the wired Internet is primarily due to its layered architecture [17], the case for cross-layer networking in wired networks is less clear than it is for wireless networks. However, the authors of [20] contend that cross-layer associations are critical to network management tasks in the wired Internet, such as backbone planning, maintenance, and failure diagnosis. A cross-layer policy server and cross-layer database are proposed that would store the cross-layer associations between layers and enable the use of management applications that would exploit this information. In this paper, the authors focus on a discussion of the architecture of this cross-layer service.

3 Cross-Layer Network Stack Design & Architecture

This section discusses how we achieved X-Layer's cross-layer network stack according to the model outlined in Figure 1. We discuss the pertinent protocols that were implemented at each layer, as well as the key variables that were exposed to cross-layer adaptation. Base case protocols were developed for each relevant layer conforming to an independently layered design, while their cross-layer cousins were developed in parallel by modifying the base cases to allow for cross-layer adaptation to other layers' parameters.

The choice of each protocol was directed by experience in sensor network design. For instance, the routing layer is a "tree" protocol which is a natural protocol for many sensor network applications. Many sensor network applications require data from many nodes to be routed to a "root node" or "sink node". The nature of networks in which many nodes within a network are spread throughout a system of interest and one node is connected to a storage system lends itself to tree routing protocols. The MAC protocol is partially integrated into the Chipcon CC2420 radio, which provides link layer ACKs (important for reliability of sensor data) while using relatively low power to operate the radio (compared to 802.11 or bluetooth wireless systems). Frugal power usage is a key to making low cost sensor systems last in the field as long as possible. Also, since low power is important the transport layer tries to achieve reliability without using too much power or being too complex.

Our cross-layer network stack is implemented using standard WSN hardware, namely TELOSB motes [21], and standard sensor OS software.

3.1 MAC & Physical Layer

The physical layer and its medium access control (MAC) layer present us with an interesting set of possibilities for cross-layer design. These optimizations or shared variables center around trading performance for power, or vice versa. Performance can be degraded, both in terms of radio power (wireless range) and MAC layer transmit/retransmit rates, in order to use less power and fewer bandwidth resources.

The possibility for cross-layer integration of the MAC layer depends on the particular protocol being used. However, many MAC layer protocols support some form of beacons and/or retransmits (auto-acknowledgements). For this cross-layer design both were supported and the routing layer took advantage of the "link-level" auto-acknowledgements built into the ChipCon CC2420 radio. It is important to note that in this case the physical and MAC layers have already been partially interwoven due to the use of a radio that includes some MAC layer functionality.

Adjusting radio transmit power is a possibility whenever the particular radio and driver software supports an interface for setting radio transmit power. In our specific implementation the ChipCon CC2420 radio was used and allowed us to modify radio transmit power. This functionality was integrated into the design of the routing layer.

Other possibilities for cross-layer integration of the MAC layer could include modifying the maximum number and rate of MAC layer retransmissions, and using the link quality metrics (RSSI or LQI) directly (in this case the routing layer uses these metrics for another metric ETX).

3.2 Network Routing Layer

The protocol built for X-Layer's routing layer is the Bidirectional Tree Protocol (BTP), a close relation to the Collection Tree Protocol [22] (CTP). BTP extends CTP to support bidirectional unicast routing, i.e. routing in both upstream and downstream data directions. The original CTP design forms a routing tree centered with a base station as the root or data sink. Unicast messages are routed towards the root by the tree's leaf and branch nodes. As a basic tree protocol, CTP is based around a distance measure, specifically the distance or depth from a root. This distance measure is used to guide packet routing toward the root node. Such tree protocols are especially well suited for "data-centric" applications common to WSNs. BTP leverages this tree network topology to perform unicast routing in the reverse or downstream direction, i.e. from base station to a leaf sensor node.

BTP, like CTP, is best used for relatively low traffic rates. It is best effort, meaning that it does not guarantee 100% delivery (this makes it a good protocol to use with the transport layer). Also, BTP makes the assumption that it can in some way infer the link quality of neighboring nodes in the network (those nodes that are only one hop away). This link quality is used to order the tree and make routing decisions. BTP assumes that nodes use link-level retransmissions (which is true of the 802.15.4 MAC layer in this case).

The most important value or variable that guides the organizing and routing within BTP is called ETX [23], the estimated number of transmissions necessary to get to the root or to get to certain other destination nodes in the network. ETX is cumulative and therefore implies the total number of transmissions that will be made on all hops along the path taken from origin to destination. ETX is used by BTP as a "routing gradient" meaning that packets will follow a path of least ETX. In our BTP implementation, we obtained the CC2420 radio's link quality indicator (LQI), rather than RSSI, as an estimator of the link quality [24]. We defined ETX as a sum of all LQI values from each hop from a source node to a destination node. A weighted moving average algorithm is used to smooth changes to LQI over time. In our implementation, LQI is scaled such that LQI is inversely proportional to link quality. That is, large LQI values indicate poor link quality, while small LQI values indicate good link quality.

```
if (meanNextHopETX > ETX_HIGH_THRESHOLD && CURR_TRANSMIT_POWER <
    MAX_TRANSMIT_POWER) {
    increaseCurrTransmitPower();
}
else if (meanNextHopETX < ETX_LOW_THRESHOLD && CURR_TRANSMIT_POWER >
    MIN_TRANSMIT_POWER) {
    decreaseCurrTransmitPower();
}
```

Figure 2. BTP Transmit Power Adjustment Pseudocode

BTP is an example of how different routing protocols provide unique aspects of design to other layers, depending on the indicators or metrics used by the protocol. In the case of BTP, ETX is the parameter that is exposed to other layers as an indicator. The method by which we exposed ETX was to create a special API function that could be called to read ETX by any layer. The ETX value is used by other layers to infer connectivity over the implied topology. Therefore, it is used to modify the radio transmission power, which in turn can result in a different topology that often leads to better connectivity (it did in all our tests, and is rather intuitive).

The routing layer provides us with an interesting chance to integrate cross-layer design with multiple layers both above and below. In X-Layer, the routing layer provides metrics for the application and transport layer while incorporating variables from the physical layer into its design.

The cross-layer behavior implemented for BTP involves an interaction with the physical layer. BTP adjusts radio transmit power for each wireless sensor node in the network based on the ETX value for the next hop in the route toward the destination node. A weighted moving average algorithm is used to smooth changes to next-hop ETX over time. If the smoothed next-hop ETX value exceeds a fixed threshold value, then we increase transmit power. Since high ETX values indicate poor link quality, the goal here is to improve link quality by increasing transmit power. If the smoothed next-hop ETX value is below a minimum fixed threshold value, then we decrease transmit power. Since low ETX values indicate good link quality, the objective of this logic is to reduce transmit power and thus reduce energy consumption while still maintaining good link quality. Figure 2 shows a representation of this logic in pseudocode. By running this transmit power adjustment logic in a distributed fashion on all sensor nodes in the network, we seek to find a near-optimal logical network topology and route between the source and destination nodes.

```
currETX = (ALPHA * newETX) +
          (1 - ALPHA) * averageOfPreviousFiveETXs;
ETXdiff = newETX - currETX;

if (ETXdiff > 0)
      increaseSendRate;
else if (ETXdiff == 0)
      doNothing;
else if (ETXdiff < 0)
      decreaseSendRate;
```

Figure 3. Application Send Rate Adjustment Pseudocode

3.3 Transport Layer

X-Layer implements a simple stop-and-wait reliable transport protocol that provides support for multiple concurrent connections between two nodes in the network, called the Lightweight Transport Protocol (LTP). Stop-and-wait was chosen instead of a windowing scheme to reduce memory usage and code size, which are of particular concern on resource constrained wireless sensor nodes. End-to-end ACKs are used to ensure reliable data transfer between the sending and receiving nodes. We use a timeout mechanism based on the Jacobson-Karels algorithm for the base protocol.

The cross-layer behavior implemented for the transport protocol involves an interaction with the routing layer. This cross-layer behavior makes use of ETX information provided by the routing layer (BTP) to set the timeout in the transport layer. Since ETX is cumulative in BTP, we use the ETX from the source to the destination node to estimate the number of hops between the source and destination by dividing ETX to the destination by an ETX per hop value.

$$numHops_{estimated} = \frac{ETX_{cumulative}}{ETX_{per\_hop}}$$

Based on experimental data, we have observed that an ETX value of approximately 12 indicates good link quality over one hop for our system, and is thus used as the ETX per hop value. Logic in LTP tracks changes in the estimated number of hops from the source to the destination over time. If the change in estimated number of hops over time exceeds some HOP_COUNT_DELTA threshold, then we use the estimated number of hops to set the timeout value in LTP. Based on observations of our system, we have found that 1000 ms is a reasonable value for timeout over one hop. Therefore, to set the timeout value, we simply multiply the estimated hop count by the timeout value per hop (1000 ms).

$$timeout_{end-to-end} = numHops_{estimated} * timeout_{per\_hop}$$

By using ETX to the destination to set timeout, we are able to adapt more quickly to dramatic changes in connectivity between the source and destination nodes than would be possible with Jacobson-Karels.

3.4 Application Layer

We designed a basic sensing application that produced sensor data at a fixed rate of one packet every five seconds. This data application was not adaptive to changes in network topology or link quality. This is typical of many deployments of sensor applications [25, 26].

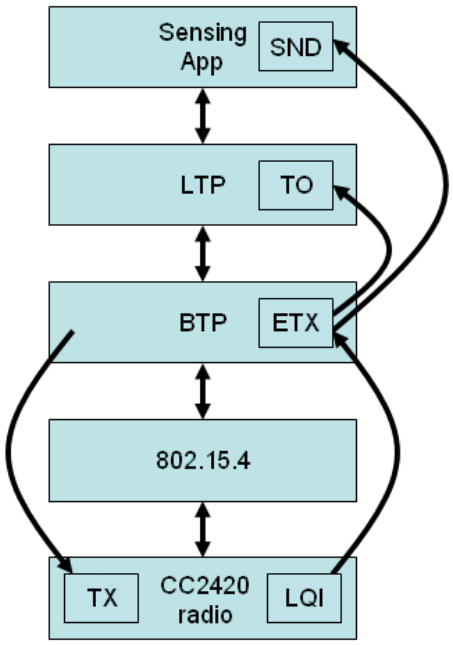

Figure 4. Summary of specific cross-layer interactions implemented in X-Layer.

To explore the effect of cross-layer interactions, we modified this application to adjust its packet sending rate in response to changes in network quality. In particular, we explored the cross-layer interaction between the application layer and the routing layer. The application receives link quality information from the routing layer and uses that information as an input to improve the overall throughput as well as reduce current draw.

Our sample application received the ETX values for each packet from the routing layer, performed a weighted averaging algorithm with the previous values, compared the smoothed result with the current ETX average, and adjusted the packet send rate based on the comparison. If the new ETX average is better than the current ETX average, the node takes advantage of the good link quality and sends packets at a higher rate. On the other hand, if the new ETX average is worse than the current ETX average, the node assumes the link quality has dropped and sends packets at a lower rate. Figure 3 represents the pseudocode for this logic. The ALPHA parameter could be used to adjust the weights of the averaging algorithm so that the ETX value does not fluctuate as much when the wireless link is unstable. How much the rate is adjusted varies from environment to environment.

3.5 Summary of Cross-Layer Interactions in X-Layer

Figure 4 summarizes the pairwise cross-layer interactions that were implemented in X-Layer. BTP calculates ETX values and exposes them to both the transport and application layers, which adjust timeouts and sending rates respectively. BTP queries the physical layer to receive LQI information and also adjusts transmit power dynamically.

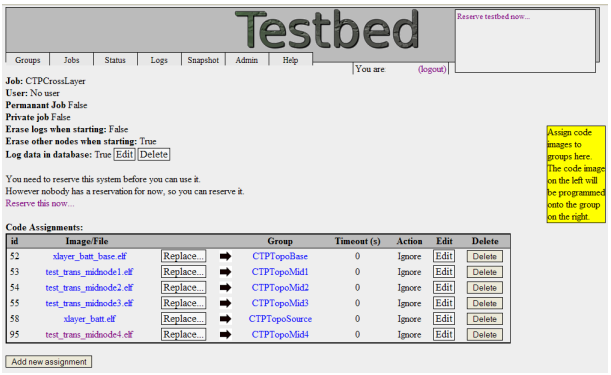

Figure 5. Testbed Programming and Data Collection Software

4 Cross-Layer Evaluation

This section explains the methodology and hardware used to evaluate our X-Layer network protocol stack. It discusses the experimental setup and comparison cases. The results from the experiments are presented in the second part of this section. Finally, we present observations that lead to a new cross-layer architecture which we feel more fully realizes the potential of cross-layer networking in wireless environments. This solution addresses how a cross-layer network stack may be designed in an orderly abstract manner while managing shared resources in an efficient and organized way.

4.1 Experimental Methodology

We evaluated the performance of our X-Layer cross-layer network protocol stack and compared it to an independently layered network protocol stack. This comparison was performed over an indoor testbed of 56 TELOSB motes. The testbed was set up to allow download of program images via USB cables attached to each mote. We developed the software to manage the testbed for our experiments. Figure 5 shows a screen shot of the testbed collection software's programming and data collection environment, in which the ELF files corresponding to our network stacks were loaded along with an end-to-end communication application onto the sensor nodes in the testbed. Depending on the power level, the network topology varies from an all-to-all network, in which every node has a one hop connection to every other node, to a multi-hop network sometimes four hops across. Diagnostic output was collected via the wired USB infrastructure while the network stacks executed and forwarded wireless data.

We evaluate three different non cross-layer "basecases" for comparison to our cross-layer stack. The basecase code runs the same protocols at each layer as our cross-layer design: LTP transport, BTP routing, and the same application code. The three basecases are configured to different radio power levels corresponding to different network topologies. The power levels given to the CC2420 radio interface are 2, 10, and 30 (about dBm -2.87E+05, -1.10E+05, and -9.14E+02 respectively). We will refer to these three cases as basecase.2, basecase.10, and basecase.30. We compare these three cases to the X-Layer cross-layer network stack implementation.

We also conducted another comparison between two different network stacks running beneath application code that attempts to extend battery life through reducing the application layer send rate as battery voltage drops. The performance of differently configured network stacks are compared to each other in order to understand how optimizations in one layer (the application layer) may be affected by configurations within other layers (the routing and transport layers). The methodology for this test was to run the same energy-aware adaptive application on top of the basecase.2 stack and also the basecase.10 stack. These network stacks were chosen for the differing topologies they provided, a good example of how small changes in network stack configuration affect the performance of other layers.

For analysis, we focused on three metrics: **goodput** (both in absolute terms and as a success ratio of packets sent from the application layer), **total throughput** (total amount of bandwidth put in the air by the network), and **latency** (end-to-end time for application layer to send and receive). "goodput" will give a sense of how much data is actually being sent through the network successfully and when compared to total throughput one can see how efficient the particular network stack was at achieving that "goodput". The latency is particularly important for understanding how the changes to transport layer timeouts and resends affects the overall end-to-end network latency.

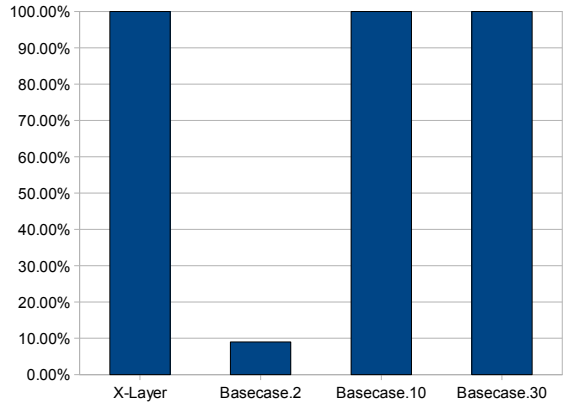

Figure 6. End-to-End Receive Ratio

4.2 Experimental Results

The first metric presented is that of end-to-end receive ratio. This part of the evaluation is meant to show what overall performance the compared cases achieve. These results should be kept in mind when evaluating the other performance metrics. In Figure 6, we show the percentage of packets that are successfully transported end-to-end from source node to the sink, or root node of the tree. The experiments demonstrated that given reasonable connectivity within BTP, the transport layer ensured 100% packet delivery throughout the length of the experiment for both X-Layer and most variants of the discretely layered stack. However, in basecase.2, in which the transmit power is held at a constant setting of 2, the connectivity was intermittent and resulted in

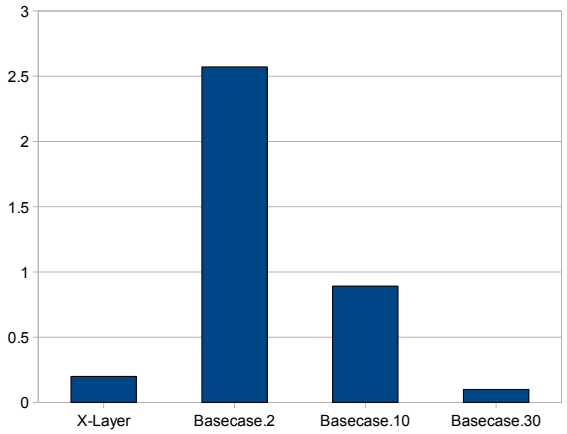

Figure 7. Resends Per Packet Sent

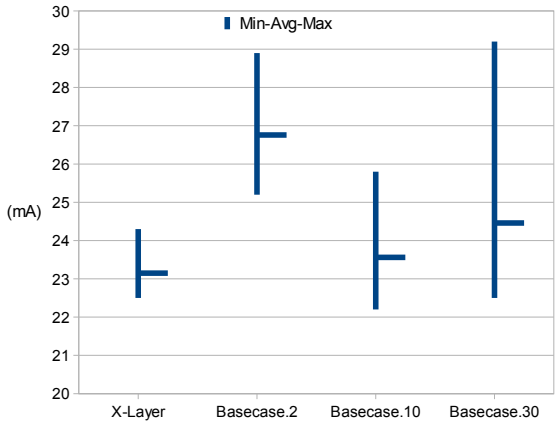

Figure 8. Minimum/Average/Maximum Power Usage (mA)

a very low receive ratio. However, poor connectivity was not only due directly to low radio transmission power, but as will be shown later in the discussion of Figure 7, the poor connectivity resulted in a greater number of retransmissions (more wireless traffic) and likely caused interference. When running protocols that retransmit or resend failed packets, poor connectivity can leads to more transmissions and more inter-node interference, further degrading end-to-end network connectivity.

This interference is evaluated through measuring the total number of resends per packet so as to understand the amount of bandwidth actually being sent by the network in order to achieve the end-to-end success rates from Figure 6. Figure 7 shows for each test case the average number of resends per packet, which is inversely proportional to goodput. Specifically this metric refers to the average number of *extra* transmissions (optimally, all packets would need only be transmitted once by the sender). The best performer in this case was the non-cross-layer stack set to a constant transmission power of 30. This is not unexpected due to the fact that this power level results in an all-to-all network topology with very good connectivity. However, this is achieved at the cost of much higher energy usage by the radio. The cross-layer stack, results in a slightly higher number of resends than basecase.30, however it is still much more efficient than both basecase.10 and basecase.2. While basecase.10 had sufficient inter-node connectivity to successfully transport all packets across the network it resent packets more than 8 times as often as basecase.30. As stated earlier, the high number of retransmissions in basecase.2 further degraded the already poor inter-node connectivity.

The performance of the cross-layer stack in terms of retransmissions is rather impressive considering that the cross-layer network code resulted in a changing topology, optimizing radio transmission power for multi-hop or single-hop topologies depending on inter-node connectivity. In spite of the changing topology the cross-layer stack was able to fix routes by modifying transmission power quickly using information from the MAC and routing layer. This information was concurrently taken into account at the transport layer, which could then modify its connection timeout so as to avoid unnecessarily resending the packet. This is an example of how sharing of information between the layers made all layers more resilient to sudden changes within each other.

While three of the cases (basecase.10, basecase.30, and X-Layer) achieved 100% end-to-end receive ratios (Figure 6) the costs in terms of total transmissions (or traffic created) were not the same. Neither was the amount of power the different cases required to achieve the same end goal, as presented in Figure 8. The average power in terms of current consumption was measured every 10 seconds while the experiment was running. The minimum, mean (average), and maximum of these power usage values are shown for each case in Figure 8. While basecase.30 achieved the best connectivity and transmission efficiency due to higher radio transmission power, it was precisely the high transmission power that caused this code to use more overall power than both the cross-layer code and basecase.10. Even though basecase.10 ended up resending packets more than 8 times as often as basecase.30, it still used less power on average to achieve the same goal of 100% end-to-end packet transport. It should be noted that because basecase.10 made many retransmissions, continuing to increase the application layer send rate may have resulted in basecase.10 retransmissions causing enough interference so as to cause the network to break down, while basecase.30 may have continued to support a higher application layer data rate. However, for many sensor network applications a send rate of 1 packet every five seconds is more than enough [27, 28]. The basecase.2 experiment showed how detrimental it can be to have radio transmission configured too low. If the transport or routing layers are retransmitting packets too often due to bad connectivity, then much more power can end up being consumed as presented in the relatively higher average power usage of basecase.2. This is a classic case of how locally designing or optimizing certain layers can affect other layers in a very negative, and often perplexing (or contradictory) fashion. In this case, lowering the physical layer transmission power, perhaps to save power, resulted in the opposite

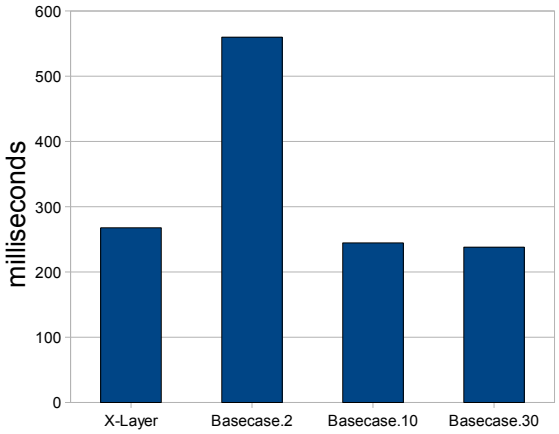

Figure 9. Average End-to-End Latency

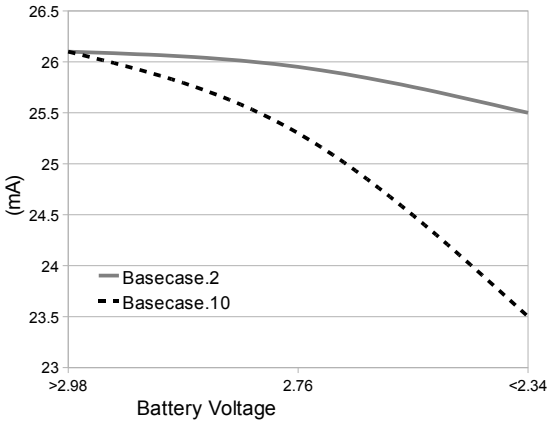

Figure 10. Energy-Aware Application Layer Send Rate Adaptation

effect, consuming more power overall due to transport and routing layers creating extra traffic to compensate for poor link connectivity. In the end, the overall system used much more power than the same code with the radio transmission power turned up (and failed to achieve reliable transport).

The cross-layer stack uses less power on average than the other cases. This is due to X-Layer implementing the transport and routing layers to react to changes in other layers. High connectivity was achieved while keeping transmission power low. The optimization of competing factors (power usage and connectivity) was not explicitly written into the cross-layer design. Rather, each layer was designed to react to information from other layers in a reasonable pairwise fashion. When connectivity is bad on a packet arriving at physical layer, the routing layer would immediately react by changing the routing tables to reflect this fact, changing the ETX value. Also, the transport layer may adjust its timeout value. All these changes allowed for connectivity to be established while keeping power low and avoiding many unnecessary retransmissions. Due to the cross-layer design, reliable transport was achieved efficiently (few transmissions) using the lowest average power usage.

The final metric evaluated in Figure 9 is that of end-to-end latency, namely how long it took a successfully relayed packet to travel from the source node to the root. This metric is most reflective of the topology of the network. In basecase.10 and basecase.30 there was always a direct link (of varying quality) being used by the routing layer, hence the latency is reflective of how long it took on average to send, receive, and acknowledge a single transmission. In the basecase.2 experiment it usually took 3 or 4 hops for packets to reach the root from the source and this is reflected in higher average end-to-end latency representing many transmissions within the network. In the cross-layer experiment the topology favored a direct connection (when good quality) between the source node and the base. However, sometimes transmission power was reduced and a multi-hop network used. This changing topology is reflected in the slightly higher average end-to-end latency of the cross-layer experiment.

4.3 On the Benefits of Cross-Layer Visibility to Applications

In this section, we explored the benefits of cross-layer visibility to application designers. We constructed a simple proof-of-concept test to show how a typical application might benefit from monitoring and reacting to lower layer system information. In this particular case the application was written to extend sensor life by reducing packet send rate as the battery voltage dropped. For certain applications this may not be acceptable, while for others lowered fidelity is tolerable. Our goal was to investigate the benefits of taking a complete system approach towards development at each layer. The test was conducted separately from other tests, and compared the life of nodes sending at a constant rate and nodes that adapted their send rate to battery voltage. While it is rather intuitive that reducing send rate would reduce the overall power usage, some interesting behaviors were observed and are presented in Figure 10. The send rate was reduced gradually from one packet every 5 seconds down to one packet every 10 seconds (resulting in halving the frequency of sensor readings) depending on the battery level. This was meant to simulate a sensor network that is still trying to achieve reasonable frequency of reading while extending battery life.

We observed that energy-aware adaptation by reducing the send rate at the application layer was not always effective, depending on the underlying network conditions. Adaptation was relatively ineffective while running the basecase.2 network stack, i.e. the overall power usage did not drop significantly despite reducing the send rate at the application layer. This is probably due to the fact that end-to-end message delivery would often fail and result in many retransmissions and transport layer resends within the network. The initial application layer send rate was already greater than the network stack could handle and as such, lowering the send rate resulted in small changes in overall power consumption. However, when running the application code above a more highly connected and reliable network, as in basecase.10, the reduction of application send rate did result in 4 to 6 times

greater power conservation. It is interesting that a rather obvious power conservation technique (reducing application send rate) may or may not have the desired effect of saving power, depending on the configuration of the network stack. This empirical result confirms that performance of the overall system can benefit by extending cross-layer visibility of the network stack all the way into the application layer. Also, it shows how the lack of information from other layers could lead to optimizations having little or no effect. Basecase.2 lost half of its sensor readings and saved very little power in the process. If information about network connectivity and the topology had been taken into account at the application layer this wasteful "optimization", which resulted in an ineffectual degradation of quality, could have been avoided.

4.4 Inter-Layer Resource Broker: A New Mechanism for Managing Network Layer Interaction

We extract several observations from our implementation experience with X-Layer, in particular some of its shortcomings that help lead us to an improved cross-layer design.

Observation #1: X-Layer is limited in its ability to provide full N-layer joint optimization of policy behavior.

While our implementation has quantified the benefits of pairwise cross-layer interaction, our observation is that a mediating entity would provide a more powerful mechanism for coordinating policy decisions, e.g. don't unnecessarily reduce sending rate as in the previous subsection, and arbitrating conflicts between adaptation policies of differing layers. In the most general case, such a mediator would arbitrate the possibly conflicting policies of all N layers, developing jointly optimized policy behavior. While our results did not indicate any cases where policies were in direct conflict, we feel that we should consider the case where adaptation policies come into conflict, e.g. one layer raises the transmit power while another wants to drop it, or one layer increases data rate while another layer wants to drop it. Even worse, it is possible to create a feedback loop [17], wherein one layer adjusts a parameter, causing notification to another layer to adjust another parameter, causing a domino effect of adjustments that may be endless. This feedback may be unstable, resulting in adaptation that amplifies poor performance. By instituting a centralized mediator that understands the individual adaptation policies of each layer, then the mediator can arbitrate between policies, decide on a best course of action that reconciles possibly competing policies over all layers, and jointly maximize the desired system metric. At the same time, the mediator may forestall any feedback loops within the cross-layer stack. For example, if each layer is reading the battery level and employs its own independent energy conservation policy, then the resource broker can realize that the application layer is backing off too harshly in its sending rate and should keep it as is, while the routing layer is right in its rate of backing off on extraneous transmissions. Thus the resource broker can adjust what were formerly independent energy backoff policies into a jointly optimal policy.

Observation #2: A more general and efficient sharing mechanism, e.g. a generic publish/subscribe interface, is needed for cleanly sharing parameters, metrics, and data structures and for efficiently notifying interested layers of updates in shared information.

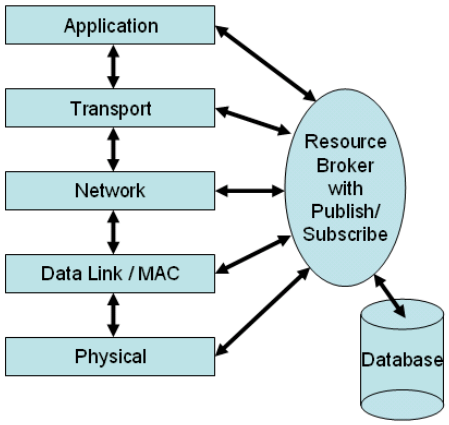

Figure 11. Informed by our experiences with X-Layer, an outcome of our research is a new cross-layer architecture consisting of a resource broker that (1) jointly manages and resolves policy conflicts, and (2) provides a generic publish/subscribe mechanism for sharing data between all layers.

Our experience with X-Layer indicated that the sharing of parameters, metrics, and data structures between two particular layers was largely ad hoc and specific to those two layers. The sharing of each variable between a given two layers required detailed semantic knowledge of each layer, i.e. where in a layer's code to read/write a given variable that was being exposed to another layer. When seeking to share information between any two layers, we face a combinatorial explosion in the amount of insider information required to specify the location and semantic meaning of each shared data entity. As a result, it is incumbent on our design to develop a cleaner, more efficient, and more general mechanism for sharing information. In addition, our current implementation essentially polls the shared information. A more efficient implementation would employ a callback to notify interested layers only of sufficiently interesting new shared information, i.e. those that meet a filtering condition specified by each layer. For example, such a notification engine could inform a layer L only if the link quality metric has dropped below a threshold specified by L.

Given our experiences and observations above, an outcome of our research is the cross-layer architecture shown in Figure 11. This architecture is able to more fully realize the vast potential of cross-layer network protocol design while doing so in a manner that is clean and general. This architecture for a cross-layer network stack consists of an *inter-layer resource broker* and a database-like structure for storing shared information between layers. The resource broker employs a publish/subscribe mechanism to share both data packets and control parameters between each of the layers. In addition, the resource broker mediates or resolves differences in adaptation policies between each of the layers.

The publish/subscribe mechanism is used to publish and subscribe to metrics and protocol parameters calculated by each of the layers. For example, layer 5 may calculate a performance metric of use to layers 2, 3 and 4. Layer 5 can publish this performance metric as often as it desires, and each layer may specify a predicate upon which they are independently notified once the predicate is satisfied, e.g. layer 4 wants notification only if the metric exceeds some threshold, layer 3 desires to be notified only if the magnitude of the change exceeds a second threshold, and layer 2 indicates to be notified of any changes at all in the metric.

We invest a further degree of intelligence into the resource broker so that it influences the adaptations that occur in the individual layers, e.g. rate and energy adaptation policies, by monitoring and controlling the publishing of values and notifications. The resource broker develops a jointly optimized adaptation policy, let's say for rate adjustment as a function of link quality and battery level, and only permits layers to adapt in response to the joint policy. The resource broker can control what each layer publishes and sees, and in this way can influence the adaptation policy of each layer, e.g. slowing the rate backoff in layer J while accelerating the rate backoff in layer K. In this manner, the resource broker is able to control and forestall any of the feedback loops that might lead to instability or endless adaptation [17].

Our resource broker architecture compensates for many of the limitations in the original pairwise cross-layer design shown in Figure 1, as discovered through our X-Layer implementation. The publish/subscribe mechanism provides a clean way to specify shared data structures between any layers, without requiring in-depth understanding of the code in each layer, hence the location of shared variables. The subscription mechanism provides a far more efficient way to notify interested layers that a condition/predicate has been met, in comparison to polling. Moreover, the architecture enables comprehensive N-layer joint optimization of cross-layer network adaptation policies, while also removing looping effects from the adaptation.

Another key advantage of the resource broker architecture is that it effectively solves the combinatorial explosion problem experienced in pairwise cross-layer interaction. The complexity of the resource broker architecture is linear with the number of layers N and shared data structures, whereas the complexity is proportional to N^2 in the pairwise cross-layer architecture.

Note also that the new architecture subsumes the original one that was investigated in this paper. That is, every pairwise interaction described in Figure 1 can also be duplicated in the new architecture via the resource broker and database.

We believe that such an architecture is feasible on WSN hardware and software. We estimate that such an implementation will take 1 KB, with the pub/sub mechanism taking 100 bytes, the database taking 400 bytes, and the resource broker's joint optimization mediator taking 500 bytes. Our preliminary work in this area has implemented a shared memory segment that can be used to share information between layers.

Our architecture bears some resemblance with the cross-layer model proposed in Kompella et al [20], but there are key differences. Kompella et al are motivated by a desire to diagnose wired IP enterprise networks. In their design, lower layers reveal their status to a policy server, which stores the information in a database that can be queried by higher layer management applications. In this way, applications achieve cross-layer visibility into the lower layers. However, Kompella's model views the information flow as essentially one way from lower to higher layers, whereas our architecture is more flexible and allows bidirectional sharing of information between any two layers. Moreover, the publish/subscribe mechanism is missing from Kompella's model. Also, their policy server appears to be primarily an information conduit that lacks the intelligence to jointly optimize and resolve conflicts among the adaptation behaviors of the individual layers, unlike our resource broker.

5 Discussion

Sensor networks can make very few assumptions about the underlying connectivity. As well, connectivity can vary to a great degree. Overdesigning each layer to ensure reliability is a very expensive option for WSNs, in which doubling performance may require more than double many other resources, increasing cost to a level that renders the sensor network impractical and sensor nodes large and fragile. Sensor networks impose tight constraints on the design of protocols and often leave little room for waste. As such, more complicated cross-layer designs at the software level may result in a working system, which would otherwise not be functional and operational. As such, integrating network design and optimization is much more important within the network design scope of WSNs. This paper shows how cross layer design can use resources much more efficiently and proposes a design system which enables a new type of abstraction (brokers) allowing for network layer abstractions to be eased. As well, the cross-layer stack showed more resilience to changing topologies, confirming our belief that cross-layer themes are a promising direction for WSN systems research, and giving us hope that such cross-layer network stacks may be more suitable for many more types of sensor networking applications than traditionally designed network stacks.

Finally, it was chosen to evaluate and present the cross-layer experimental results as a whole rather than in pieces due to the strongly interwoven nature of the different layers in the cross-layer stack. This interwoven property should be stressed as a key property of conventional cross layer design and is a major motivation for introducing the concept of the resource broker cross-layer architecture. The abstraction between layers should not lead to each layer's design being completely independent from the other layers. This paper shows how modifying or optimizing one layer without taking other layer designs into account can lead to waste or contradictory system behavior due to inter-layer interaction. As such, X-layer was designed with each layer in mind and evaluated as such. X-layer is not just a network stack that shares information between layers, but a network stack designed in a cross layer fashion, that is, the network layers were designed to interact with each other as an integrated whole. This approach showed the greatest improvement and most efficient usage of resources.

6 Future Work

We plan to implement in more detail the generalized version of our cross-layer network architecture shown in Figure 11. We have already begun by implementing a shared memory segment that can be used to share information between layers. In addition, we have protected access to this shared memory with synchronization semaphores to ensure atomic reads and writes. This will serve as the basis for our database. On top of the database, we will need to incorporate a publish/subscribe subsystem that allows a layer to specify a predicate or condition for notification. We expect that such a language should be relatively basic, allowing for logical AND's and OR's as well as threshold comparisons, e.g. *if ((X-Y)>Z && |M[i]-M[i-1]|==12 || U*V<7) then notify Layer L.* We hope to be able to incorporate elements from existing pub/sub systems, e.g. SIENA [29] and READY [30].

Once the resource broker architecture is functional, we intend to explore the space of joint optimizations that may be achieved to yield the most marked improvement in rate adaptation and/or energy adaptation. For example, maximizing throughput comes at a cost in energy, whereas maximizing energy lifetime can be achieved by sending no packets (zero throughput!). Clearly, there are several optimization functions that we can formulate: optimize throughput subject to energy lifetime greater than some minimum threshold; maximize energy lifetime subject to throughput exceeding another minimum threshold. We are not sure which formulation would yield the highest satisfaction to the WSN end user, and the answer may well be domain specific. Given a formulation, we would need to evaluate the extent of the advantage of X-Layer in meeting the metric compared to independently layered solutions. We would further like to perform these evaluations in an in situ deployment. We would also like to study the role of our resource broker in mitigating instability and endless adaptation by neutralizing feedback loops.

We will seek to explore in more detail which set of pairwise interactions enables the greatest improvement in performance. Our X-Layer efforts have demonstrated that there are meaningful improvements to be achieved from a specific set of pairwise cross-layer interactions, but we have not yet answered which sets of pairwise interactions gives us the biggest bang for the buck. We believe this is a rich area for further research.

Our X-Layer research also has not sought to fully minimize memory usage by completely exploiting the overlap between functionality, buffers, and data structures shared across several network layers. We will seek to counterbalance the hopefully modest increase in memory footprint due to added resource broker functions with a corresponding decrease in memory consumption due to more active sharing of data structures and functions across layers.

We intend to release the X-Layer network stack as open source code to provide a platform upon which other researchers can build more highly optimized versions of this cross-layer network stack.

7 Conclusion

We have described in this paper a detailed implementation of a real-world functioning cross-layer network protocol stack called X-Layer on standard WSN hardware and software. We have shown performance improvements of X-Layer compared to three base cases of a traditional discretely layered network stack. Our experimental evaluation on a testbed of WSN motes has shown that X-Layer achieves the best balance of network throughput, goodput, energy usage, and latency compared to independently layered designs. Most importantly, good cross-layer design can avoid waste and inter-layer thrash in already resource starved sensor network systems. In every one of the metrics evaluated, the cross-layer stack performs close to or exceeds the best performance offered by any of the variations of the traditional stack. Informed by our experiences and observations developing X-Layer, we have described a new architecture for cross-layer network protocol stacks that more fully realizes the potential gains of cross-layer designs in an elegant and clean manner. This architecture consists of a resource broker that allows all layers to share information via a generic publish/subscribe interface. This resource broker is further imbued with the intelligence to resolve potentially conflicting rate/energy adaptation policies and develop a single jointly optimized policy that spans the heterogeneous layers.

8 References

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

[2] V. Raisinghani and S. Iyer, "Cross-layer design optimizations in wireless protocol stacks," *Computer Communications*, vol. 27, no. 8, pp. 720–724, May 2004.

[3] S. Shakkottai, T. Rappaport, and P. Karlsson, "Cross-layer design for wireless networks," *IEEE Communications Magazine*, vol. 41, no. 10, pp. 74–80, October 2003.

[4] X. Lin, N. Shroff, and R. Srikant, "A tutorial on cross-layer optimization in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 8, pp. 1452–1463, August 2006.

[5] A. Dunkels, "Full tcp/ip for 8-bit architectures," in *Proceedings the First International Conference on Mobile Systems, Applications, and Services (MobiSys 2003)*, May 2003.

[6] A. Dunkels, F. Osterlind, and Z. He, "An adaptive communication architecture for wireless sensor networks," in *Proceedings of the 5th ACM Conference on Embedded Networked Sensor Systems (SenSys 2007)*, November 2007.

[7] "Zigbee specification," http://www.zigbee.org.

[8] Y. Zhang and L. Cheng, "Cross layer optimization for sensor networks," in *Proceedings of the 3rd New York Metro Area Networking Workshop (NYMAN 2003)*, September 2003.

[9] L. van Hoesel, T. Nieberg, J. Wu, and P. Havinga, "Prolonging the lifetime of wireless sensor networks by cross-layer interaction," *Wireless Communications, IEEE [see also IEEE Personal Communications]*, vol. 11, no. 6, pp. 78–86, December 2004.

[10] M. Sichitiu, "Cross-layer scheduling for power efficiency in wireless sensor networks," in *Proceedings of the Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004)*, March 2004.

[11] A. Safwat, H. Hassanein, and H. Mouftah, "Optimal cross-layer designs for energy-efficient wireless ad hoc and sensor networks," in *Proceedings of the 2003 IEEE International Performance, Computing, and Communications Conference*, April 2003.

[12] R. Madan, S. Cui, S. Lall, and N. Goldsmith, "Cross-layer design for lifetime maximization in interference-limited wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 5, no. 11, pp. 3142–3152, November 2006.

[13] J. Haapola, Z. Shelby, C. Pomalaza-Raez, and P. Mahonen, "Cross-layer energy analysis of multihop wireless sensor networks," in *Proceedings of the Second European Workshop on Wireless Sensor Networks*, January 2005.

[14] J. Polastre, J. Hui, P. Levis, J. Zhao, D. Culler, S. Shenker, and I. Stoica, "A unifying link abstraction for wireless sensor networks," in *Proceedings of the 3rd ACM Conference on Embedded Networked Sensor Systems (SenSys 2005)*, November 2005.

[15] 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, February 2004.

[16] A. Goldsmith and S. Wicker, "Design challenges for energy-constrained ad hoc wireless networks," *Wireless Communications, IEEE [see also IEEE Personal Communications]*, vol. 9, no. 4, pp. 8–27, August Aug. 2002.

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

[18] T. Weingart, D. Sicker, and D. Grunwald, "Identifying opportunities for exploiting cross-layer interactions in adaptivewireless systems," *Journal on Advances in Multimedia*, vol. 2007, November 2007.

[19] H. Aiache, V. Conan, J. Leguay, and M. Levy, "Xian: Cross-layer interface for wireless ad hoc networks," in *Proceedings of the Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, June 2006.

[20] R. R. Kompella, A. Greenberg, J. Rexford, A. C. Snoeren, and J. Yates, "Cross-layer visibility as a service," in *Proceedings of the 4th ACM Workshop on Hot Topics in Networks (HotNets-IV)*, November 2005.

[21] "Moteiv telosb motes," http://www.moteiv.com.

[22] R. Fonseca, O. Gnawali, K. Jamieson, S. Kim, P. Levis, and A. Woo, "The collection tree protocol (ctp)," Internet, August 2006.

[23] D. D. Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proceedings of the 9th annual international conference on Mobile computing and networking (Mobicom 2003)*, September 2003.

[24] K. Srinivasan and P. Levis, "Rssi is under appreciated," in *The Third Workshop on Embedded Networked Sensors (EmNets)*, 2006.

[25] G. Werner-Allen, K. Lorincz, J. Johnson, J. Lees, and M. Welsh, "Fidelity and yield in a volcano monitoring sensor network," in *Proceedings of the 7th conference on USENIX Symposium on Operating Systems Design and Implementation (OSDI 2006)*, November 2006.

[26] C. Hartung, C. Seielstad, S. Holbrook, and R. Han, "Firewxnet: A multi- tiered portable wireless system for monitoring weather conditions in wildland fire environments," in *Fourth International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2006, pp. 28–41.

[27] A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *1st ACM International Workshop on Wireless Sensor Networks and Applicatoins (WSNA 2002)*, Atlanta, GA, September 2002.

[28] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, March 2002.

[29] A. Carzaniga, D. Rosenblum, and A. Wolf, "Design and evaluation of a wide-area event notification service," *ACM Transactions on Computer Systems*, vol. 19, no. 3, pp. 332–383, August 2001.

[30] R. Gruber, B. Krishnamurthy, and E. Panagos, "Ready: A high performance event notification service," in *Proceedings the 16th International Conference on Data Engineering (ICDE 2000)*, February-March 2000.