Cooperative Cross-Layer Design for Wireless Networks

C. Hager, D. J. Shyy, and J. Ma
The MITRE Corporation
7515 Colshire Drive
McLean, VA 22102
{chager, djshyy, jma}@mitre.org

***Abstract*-Cross-layer optimization is an effective mechanism to improve wireless network performance and efficiency. Although cross-layer optimization is not a new technique, it can be applied effectively to different wireless technologies to handle different problems while still conforming to the wireless standards. Utility functions are constructed from measurements of the different OSI layers to provide input into our Cross-Layer Optimization Engine (CLOE).**

ZigBee technology was chosen as the test platform to demonstrate our proposed mechanism. Experiments were performed in simulated and live environments. Various mobility scenarios allowed observation of ZigBee devices with default operations and with CLOE. Our results show that CLOE can provide ZigBee with significant energy savings.

***Index Terms*—Cross-Layer Optimization, Energy Saving, Resource Allocation, ZigBee**

I. Introduction

There are a number of short range wireless standards that are utilized today. However wireless standards currently used in this environment are suboptimal in the areas of dynamic information technology capabilities, power consumption, survivability, and energy efficiency. Mobility also gives rise to a requirement for self-organizing ad hoc networks that are non-invasive and low-maintenance. Today's wireless sensor devices lack the ability to read and process sensor data in an efficient manner [1]. Moreover, traditional design only communicates data between adjacent OSI layers. However, cross-layer design allows optimization of relevant data across OSI layers.

This paper describes and demonstrates through experimentation a mechanism to improve communications performance and efficiency in a wireless network environment. Applications of this mechanism are designed to adapt to different situations, denoted by the environment while still conforming to the underlying wireless standards. The cross-layer optimization approach enables efficient communication of measurements among OSI layers and abstract behaviors. Before attaining such efficiency, the basic parameters exposed at each relevant OSI layer must be understood.

II. Related Work and Background

Abstract behaviors of systems can be created by means of a subsumption architecture, which was first conceptualized by Brooks [2]. A subsumption architecture is a hierarchical layering of simple reactive behaviors, and was originally designed for use in robotics. Each layer in the subsumption architecture is referred to as a *behavior layer*. Smith, *et al.* [3] use a subsumption architecture with two behavior layers to simulate a self-healing wireless network with node mobility. We extend this concept to interface an architecture with more OSI layers and allow handling of more complex scenarios.

In terms of behavior, Wolpert and Tumer [4] described *agents* that are able to adapt and learn to optimize rewards, or payoffs. The term *utility* is used to quantify agent actions, especially in terms of problem solving ability. Lynden and Rana [5] assumed that each agent will maximize their *local utility functions*, which only quantify individual agents. The *global utility function* [5, 6] would quantify the processes of an entire multiagent system. In place of agents, we apply the notion of local utility functions to individual features on a wireless node and global utility functions to the entire set of features on a wireless node.

Performance and functional utility are also discussed by Lynden and Rana [6]. Performance utility is a function of implementation and is directly affected by factors such as overhead and latency. In contrast, functional utility is more conceptually abstract and does not consider performance engineering related issues. Functional utility represents abstract behavior. In our model, each feature or application will require *measurements* from the OSI layers to create a local utility function, such that performance utility may be quantified. Merging the performance utility (essentially combining the local utility functions) can generate a global performance utility, or global utility function. This global performance would provide valuable input into what would be required for achieving global objectives in our mechanism; in other words, how much behavior activation, suppression, or inhibition would be required for functional utility in our mechanism.

We assume our system to be a dynamic wireless network, thus, our measurements require awareness of

© 2008 ACADEMY PUBLISHER

the environment context. Context is defined in this paper as information that can be used to describe the situation of an entity in the wireless network. Schilit, *et al.* [7] use "where you are," "who you are with," and "what resources are nearby" as aspects of context. Dey and Abowd [8] state that location, identity, time, and activity are the main dimensions of context. In our scheme, we use several different components or measurements of context that represent the system device, network, and environment.

Context aware applications rely on environment measurements to provide manual or automatic triggers to perform certain tasks. What sets our work apart from most context aware applications is that we utilize the context measurements for cross-layer optimization to create an efficient architecture. Our mechanism requires measurements to provide itself with relevant context information to perform its behavioral objectives. In other words, the measurements are essential to make decisions or react to different situations. Examples of other context aware applications and systems are discussed in references [7-10].

Sun, *et al.* [11] list some functional requirements for the design of a context aware system. These requirements include context collection, context storage and management, context subscription and delivery, and context analysis and composition ability. We apply these functional requirements to the design of our mechanism.

III. DESIGN

This mechanism was developed to improve network performance and efficiency in wireless environments. A potential goal could be to provide devices in the environment with improved performance and energy efficiency while maintaining desired behavior, or other objectives.

We propose to have one Cross-Layer Optimization Engine (CLOE), the software, to reside in each wireless node. It performs cross-layer optimization using measurements from different layers. Each CLOE formulates an optimal decision based on various parameters. CLOE is antenna, radio, and channel configuration agnostic.

CLOE adopts open architecture design. The design methodology follows that there would be a group of objectives associated with each OSI layer or feature. The goal of utility functions, locally and globally, is to meet the application requirements and user requirements. The applicability of the utility functions is that each OSI layer can be implemented by different vendors, and they can still be interoperable to each other. The application requirements can include maximizing throughput (capacity) or minimizing delay (performance). The user requirements can include minimizing power consumption (resources) or providing stable routes.

The architecture of CLOE is divided into two modules. These modules include the measurements gathering and decision making processes. First, the measurements module collects information from the surrounding environment. Some measurements are stored and are updated every time new measurements are gathered. The output of the measurements module is used for the decision module. The decision module contains algorithms and utility functions, that are used for establishing certain behaviors, leading to efficient applications.

Our design utilizes the "knobs and dials" (parameters) that are exposed to CLOE. As a result, the "spaghetti design" issue [12] does not apply to us. The OSI layers are intact and CLOE is designed to operate in the background without redesigning the OSI architecture.

A. Measurements Module

The measurements module is responsible for collecting information from the environment. These measurements are quantities for use in the decision module.

1) Functionality

The measurements are used as input for the decision module. These measurements include energy, location, and network communication properties. The energy component is represented by the current battery level of the user device. The signal strength observed at the wireless network interface device affect the location and communications components.

The measurements can be checked periodically for changes or assessed on demand. On some devices, especially those with small energy capacities or low processing capabilities, there exists a significant tradeoff between periodically collecting data and only collecting data when needed. The advantage of periodic measurement collection is that the overhead can be reduced when resolving a behavior. In other words, if the relevant measurements are not likely to change within an appropriate time interval, then decisions can be made from stored measurement data without having to acquire the current context data. The disadvantage to this method is that there are periodic surges of power consumption and processing overhead when acquiring the measurements and these will result in overall reduced battery life and processing delays. Conversely, the advantage of collecting the measurements on demand potentially reduces total energy consumption, but requires additional overhead for highly reactive behaviors.

2) Measurement Components

The following are four sample components of measurements. Each component is briefly summarized.

Energy: The energy component is actually a combination of two measurements. The first is the status of the alternating current (AC) input. If the system is powered by an external power source, then, for the moment, there is no need to be concerned about remaining energy in the device and the metric value is ignored. If the AC line status is negative, then the device is solely functioning on its internal power supply and the metric is now based on remaining battery life.

Location: Location is a difficult context component to quantify with respect to behavior. Usually when people refer to location, they mean position. Locations can be

© 2008 ACADEMY PUBLISHER

inferred from positions, such as "the lobby of MITRE-Washington," "7515 Colshire Dr., McLean, VA," or "the longitude and latitude coordinates 38°55'16"N, 77°12'18"W supplied by a global positioning system (GPS) device." However, when given information such as "at home," "in the office," or "in the lab," position is not necessarily known or even needed.

Signal Quality: The signal quality component can be represented by several types of information, including packet overhead, throughput, link capacity, quality of service (QoS), and signal strength. Our design uses signal strength to determine the metric value. Signal strength will be affected if the wireless device or access point is placed near metal surfaces and solid high-density materials. If there are obstacles in the radio signal path between an access point and wireless device or between two directly communicating wireless devices, the radio signal may either be absorbed or reflected. The coverage will, hence, be decreased. In addition, other devices that share the 2.4 GHz radio spectrum, including microwave ovens, some cordless phones, and competing systems such as Bluetooth [13] or IEEE 802.11 [14], may cause interference. Low signal strength could also be a result of an intruder attempting to jam the communications link. The data rate will also drop if the signal strength is weak.

Object Size: The object size for the measurements needs to be considered. An object requiring transmission, whether it is a document, audio clip, or URL, will have a size value representing the number of bytes needed to represent and store the object. The information collected from this measurement is simply the number of bytes necessary for object storage. Large objects require a longer time to transmit than small objects; thus, more efficient behaviors are needed for larger objects.

B. Decision Module

The decision module uses the numeric data acquired from the measurements module to produce decisions. Each decision of the most appropriate functional objective, or behavior, is resolved from this module. The resulting decisions are sent as input to the OSI layers. The decisions would be the result of local utility functions and global utility functions.

1) Functionality

The decision module affects the CLOE applications and, therefore, the behavior of the device. Our decision module relies on the local utility functions on a device. Additional input would come from the global utility function of a device or interactions between global utility functions in a network. The values of the utility functions can cause activation, suppression, or inhibition of certain behaviors. For our purposes, utility functions are restricted to output values between 0 and 1. Values approaching 0 indicate poor performance, and values approaching 1 are desirable.

Local Utility Functions: The methodology used to construct the local utility functions is radio independent. The mathematical expressions of each function are specific to applications or features of the wireless device, and the measurements and *variables* (used in the utility function) are also specific to the wireless technology. The variables are the "knobs and dials" exposed from the wireless device. The local utility functions are a solution to solve the issues presented for each application. The measurements can come from all layers.

Global Utility Functions: The global utility function (cross-layer and multi-node) treats the local utility functions as tool boxes, and examines the effectiveness of each local function (i.e., it acts similarly to a learning engine). It chooses the effective local utility functions, assigns a weight to each, and combines them to achieve a global optimum. The weights can be adjusted to adapt to different requirements, scenarios, or missions. An example is a linear weighted combination of local utility functions for each application or feature. Another example would be to use a hierarchical expression, such as optimizing the utility function with the highest priority first, followed by the function with second highest priority, and so on. In this paper, we investigate the effectiveness of local utility functions. The effectiveness of global utility functions will be provided in a future paper.

2) Behavior Influence

Since the behavior layers are contained in this module it will be directly affected by local utility functions and indirectly affected by measurements. Thus, before the measurements are used, weights must be applied. These measurement weights represent the importance of a single component to the local utility function and ultimately the decision process.

The behavior layers in the subsumption architecture form their own networks of finite state machines augmented with timers, otherwise known as augmented finite state machines (AFSMs) [15]. After a period of time, a state change is possible. The AFSM input in the subsumption architecture are the utility functions. The AFSM output would affect the variables either directly or indirectly via multiplexing. Modified variables in each OSI layer would affect the system and potentially cause future measurements to change. This is how system behavior is achieved and also how feedback is provided into the system.

IV. Applications

CLOE has the following sample applications: Discovery, Association, Energy Saving, Transmit Power, Resource Allocation, Route Selection, and Cooperative MIMO. Different applications can be represented as behaviors for individual local utility functions, behaviors for combinations of local utility functions, or behaviors for global utility functions. Experiments involving the Discovery and Association applications with Energy Saving behavior are discussed below.

A. Discovery

The ZigBee specification defines two methods of discovering other ZigBee devices: IEEE address requests and Network address requests. In our mechanism, the discovery process can simply be defined as an active discovery behavior or a layered combination of passive

© 2008 ACADEMY PUBLISHER

(lower priority) and active (higher priority) discovery behaviors. In either case, the default ZigBee discovery process can be improved with cross-layer optimization, by modifying values of certain variables or working in tandem with other objectives. Variables such as *Transmit Power*, *TP*, can be maximized to increase range and *Beacon Interval, BI*, minimized to reduce discovery time. Experiments would involve multiple ZigBee devices at various ranges from each other.

B. Association

The association process would usually occur directly after discovery. This behavior can represent a device's willingness to associate with a PAN. First, ensure that the Association-Permit variable is enabled. Then, similar to the discovery behavior, modify variables to reduce discovery and association time. Since we want to utilize information from multiple layers and devices, experiments would require a ZigBee device to be particular about associating with other devices. A simple example could be denying association with a device that would transfer large files, due to a limitation in storage space.

C. Energy Saving

ZigBee has a mode for reduced energy consumption, the Battery Life Extension mode. This mode simply reduces the number of backoffs. Additional parameters that would affect this objective include reducing *TP*, or sending a route error code of "low battery level." Alone, this objective is not very interesting as it would severely limit the communication capabilities of the ZigBee devices. However, when combined with other applications, such as discovery or association, energy efficient behavior begins to emerge through the use of cross-layer optimization. Experiments can include ZigBee devices with varying notional battery levels, distances, routes, and security.

D. Utility Functions

CLOE is positioned to achieve an optimal balance among battery power, change of the local topology, frequency of performing the discovery task, and associating with the most appropriate node. In the discovery experiments, the objective is to employ the local utility function to discover a specific number of neighboring nodes, while balancing the factors mentioned above. The association experiments are an extension to the discovery experiments and we can simply use the results of the discovery local utility functions for its own utility function. Components of the discovery utility function include two variables (*TP* and *BI*) and one measurement, the *Link Quality*. To reduce the complexity of the experiments, the domains of the variable components were restricted (see Table I).

TABLE I: UTILITY FUNCTION COMPONENTS

Component	Possible Values
TP (dBm)	-30, -24, -18, -12, -6, 0
BI [a]	3, 4, 5
LQ	0 to 256

[a] *BI* times: 0.123 s, 0.246 s, 0.492 s

The *TP* variable will vary between -30 dBm and 0 dBm. The *BI* variable represents an exponent value that determines the frequency of beacon transmissions. The *Link Quality*, *LQ*, is an integer value which denotes the quality of the reception of a received frame [16]. The component utility functions can be represented as follows:

$$U_{TP} = -\frac{TP}{30\,dBm}, U_{BI} = \frac{BI-3}{2}, U_{LQ(n)} = 1 - \frac{|LQ_n - LQ_{n-1}|}{256} \quad (1)$$

The local utility function for this discovery application, U_D, could then be a linear combination of the component utility functions,

$$U_{D(n)} = w_1 U_{TP} + w_2 U_{BI} + w_3 U_{LQ(n)} \quad (2)$$

where w_k are the weights, such that the sum of the weights is 1, and n represents the number of iterations. The measurements and variable settings are entered into U_D after each iteration. An iteration represents one attempt to discover neighbor nodes, and can be performed using different methods, such as a beacon broadcast. More complex relationships between the component utility functions could be derived, but a linear combination suits our needs. The best value obtained from this function, $U_D = 1$, would be a result of finding the appropriate number of neighbors with minimal *TP*, maximal *BI,* and constant *LQ*. Maximizing U_D would utilize the least amount of energy from the device and potentially find neighbors with low mobility and relatively stable communication properties. Note that *LQ* measurements can be used to influence the values of the two variables, thus, affecting the next iteration of U_D.

With stationary nodes the problem of finding the desired number of neighbors is reduced to a stochastic process that is dependent on random variables. Given a group of nodes randomly distributed in a region, the relative distances between each node will vary according to the distribution of a random variable. However, when mobility is introduced, the acceleration vector quantity for each node complicates the discovery process. Furthermore, in different mediums the path loss and random attenuations, such as fading, need to be taken into consideration. Without *a priori* knowledge, the discovery application would need to be robust enough to manage all these different random variables.

The association utility function, U_A, can be represented by choosing the maximum mean value of U_D across n iterations (e.g., n beacon broadcasts) for N neighbor nodes.

$$U_{A(n)} = \max\left(\tfrac{1}{n}\sum_{i=1}^{n} U^1_{D(i)}, \tfrac{1}{n}\sum_{i=1}^{n} U^2_{D(i)}, \ldots, \tfrac{1}{n}\sum_{i=1}^{n} U^N_{D(i)} \right) \quad (3)$$

Ideally, the U_D for each node would return a value after every iteration. However, collisions and attenuations occur that possibly prevent the receiving of frames. In these cases, the neighbor nodes are penalized with lower values as missing frames are represented by $U_D = 0$.

V. EXPERIMENTS

This section describes our experiment plan to verify our mechanism. Each CLOE will reside in a single wireless device and interactions between wireless devices

© 2008 ACADEMY PUBLISHER

will be captured. For the wireless network devices we use ZigBee technology [16, 17]. ZigBee was chosen due to its low-power consumption, flexibility, features, and standards foundation. A background of ZigBee is given in the next subsection, followed by the experiment environment and methodology. Finally, the results and discussion conclude this section.

A. ZigBee Background

ZigBee is a wireless Personal Area Network (PAN) defined by IEEE 802.15.4 that is optimized for remote monitoring and control. ZigBee incorporates native security services and an application layer. IEEE 802.15.14 defines the Physical (PHY) and Media Access Control (MAC) layers. The Network, Security, and Application layers are specified by the ZigBee Alliance. ZigBee devices are inexpensive, small, low power and support automatic establishment of mesh or cluster networks. Security services include encryption, integrity, anti-replay, and authentication. Since these devices include an application layer, they can easily integrate with sensors to create smart wireless networks. The programmable application layer and built-in security services indicate that ZigBee devices have potential uses in many unforeseen roles. For more detailed information on IEEE 802.15.4, refer to the standard [16].

ZigBee devices come in three flavors as specified by the ZigBee Alliance [17]. ZigBee Coordinators are the most capable devices and represent the roots of network trees. ZigBee Routers are almost as capable as coordinators and are used as intermediate nodes in a mesh network. Finally, ZigBee End Devices are only child nodes that have no routing capability and usually interface with sensors.

The IEEE 802.15.4 standard [16] defines certain variables whose values can be modified by interfacing with ZigBee PAN Information Base (PIB) attributes. Commercial ZigBee devices purchased through a vendor would most likely have these variables set at the default settings indicated in the standard. Users might not have a direct interface to the PIB. In our study, we intend to utilize hardware developer kits that can enable access to the ZigBee variables. These variables are separated into PHY and MAC layer variables.

B. Live Environment

Communications were achieved via several ZigBee devices. The distance between the devices varied as required by the experiments. The devices operated in a beacon-enabled PAN in the 2.4 GHz band using omnidirectional antennas. The nominal data rates were 250 Kbps. The hardware and software components are enumerated below.

1) Hardware

One Jennic JN5121-EK000 Controller Board and four Jennic JN5121-EK000 Sensor Boards are used in the experiments. The ZigBee Jennic evaluation kit contains five ZigBee Coordinators. One device is labeled as a "controller board" and has an LCD, sensors, and four LEDs. The other four devices are labeled as "sensor boards." Each sensor board only contains the sensors and two LEDs, but can still fully function as a ZigBee Coordinator [17]. The combination of these hardware devices allows for flexible experimentation and more complex scenarios. Note that the experiments in this study are not meant to critique the vendor implementation, as our intentions are simply to investigate CLOE algorithms on an actual hardware platform.

2) Software

The Daintree Networks Sensor Network Analyzer and the C/C++ toolchain are used for the experiments. A ZigBee protocol analyzer is used for capturing and decoding data passed between devices. From an RF/Networking perspective, examination of the Physical, Data Link, and Network layers using the ZigBee protocol analyzer would help identify whether the experiments were successful.

C. Live Experiment Setup

For our experiments, we assume a relatively stable environment with negligible coexistence interference, such that while nodes are stationary, $U_{LQ(n)} = 1$. In the utility function, we used weights of $w_1 = 0.25$, $w_2 = 0.25$, and $w_3 = 0.5$ to give more emphasis on the change in LQ. We used all five ZigBee devices from the Jennic evaluation kit. The Jennic controller board represents the source node for improved logging along with one sensor board as a protocol analyzer. The neighbor nodes (the three remaining sensor boards) were placed in random locations with respect to the source node: within maximum range, out of range, uniformly distributed, and clustered. The source node would remain stationary while in the discovery and association states. The three neighbor nodes would have mobility such that the Euclidean distances from the source node are either increasing or decreasing. In other words, we assume neighbors do not orbit the source node.

We implemented the CLOE algorithm for the discovery and association processes and loaded it into the flash memory module of the source node. CLOE would employ U_D to discover a specific number of neighbors and then U_A to associate with the most appropriate one. In comparison, the default ZigBee discovery mechanism uses TP = 0 dBm, BI = 3 (0.123 s), and does not dynamically alter these values during operation. The default discovery method actually represents a subset of the possible value combinations of these two variables (TP and BI).

D. OPNET Environment

Due to limitations in the contributed ZigBee OPNET model we employed a modified IEEE 802.11 OPNET model to better simulate ZigBee communications for our experiments. We were still restricted to certain hard coded values from the IEEE 802.11 model, but for comparative purposes, the scope was sufficient. The relevant simulation parameters are shown in Table II.

© 2008 ACADEMY PUBLISHER

TABLE II: SIMULATION PARAMETERS

Parameter	Possible Values
Transmit power	-30, -24, -18, -12, -6, 0 (dBm)
Beacon interval	0.12288 s
Range	100m × 100m
Data rate	1 Mbps
Packet interval rate	Constant(0.12288 s)
Packet size	Exponential(92 bytes)
Contention period	0.06144 s
Node speed	~15 m/s

E. OPNET Experiment Setup

We created two scenarios for our simulations. Scenario 1 involved two nodes: a stationary source node and a mobile neighbor node. Scenario 2, depicted in Fig. 1, involved five nodes: a stationary source node (node 0), three stationary neighbor nodes (nodes 1 to 3), and a mobile neighbor node (node 4).

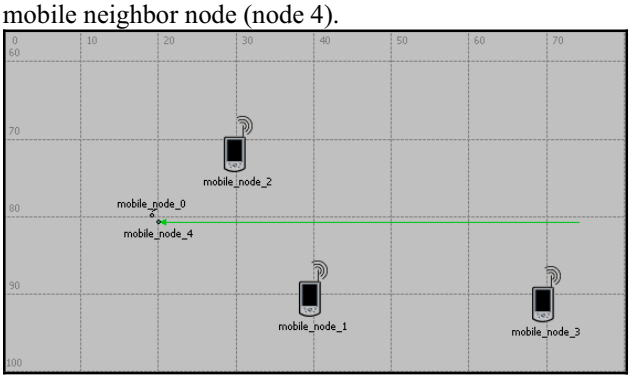

Figure 1. OPNET node diagram.

The IEEE 802.11 OPNET model source code was adapted to include the CLOE algorithms and operate as a ZigBee waveform. We utilized the signal-to-noise ratio (SNR) of received packets to represent the *LQ*. A rule for restricting SNR values between 20 dB and 30 dB was also incorporated into CLOE. The lower threshold was based on connectivity guidelines [18] for minimum node association requirements. The upper bound was included to limit excess power consumption.

VI. RESULTS AND DISCUSSION

A. Live Experiment

After a beacon broadcast from the source node, the *LQ* values from the response frames of each neighbor node in range will be inversely proportional to the distances from the source node. In other words, a high *LQ* value corresponds to a neighbor node located a relatively short distance away. Extreme *LQ* values are almost never attained. With receiver and sender *TP* values being equal, an *LQ* value approaching 20 or lower correlates to a high frame drop rate, as verified by the protocol analyzer. An *LQ* value of 255 would require the transmitting device to be located only a few centimeters away from the receiving device. Another factor contributing to a high *LQ* value is a responding neighbor with a high *TP* setting. With knowledge about the neighboring *TP* settings, mobility towards or away from the source node can be inferred after several beacon transmissions, or iterations.

1) Discovery Application

In the case where all nodes are stationary, the discovery utility function becomes solely dependent on the current *TP* setting. After one beacon transmission (n = 1), the neighbor *TP* and *LQ* values are known from the responses. After a second beacon transmission (n = 2), the discovery utility function assumes that the nodes responding are stationary, since $U_{LQ(2)}$ = 1. CLOE then maximizes *BI* and adjusts *TP* to increase energy efficiency. If more than the desired number of neighbors is discovered, then CLOE will decrease the *TP* level to remain in range of *at least* the specified number. If less is discovered, then CLOE will increase the *TP* value, until the desired number or greater is reached. In the possibility that the desired number of neighbors is never attained, CLOE will remain in the discovery state. Note that this is only done for experiment purposes to develop the CLOE discovery process. In a real operation, even if the desired number is not attained, the device will proceed to the next phase. The advantage CLOE has over the default discovery mechanism is the potential to control the reduction of transmit power and beacon transmission rate to reduce energy consumption at the device.

Experiments involving mobile nodes would employ all the components of the discovery utility function. The source node would dynamically alter its variables based on output from the utility function. Likewise, CLOE would exist on the neighbor nodes, such that they would be able to dynamically alter their *TP* settings. Default ZigBee scenarios are compared to scenarios using CLOE. We illustrate the results of a particular scenario in Figs. 2 to 4 with respect to a neighbor node. The figures show measurements of *TP*, *BI*, and *LQ* in a time series. In this scenario, the three neighbor nodes were clustered close to the source node at the beginning of the experiment. The source node was set to only discover two nodes. The cluster began moving away from the source node after a few seconds and then began moving back toward the source node after 10 seconds, reaching the source node at the end of the scenario.

At the start of the scenario, the source node initially discovers all three nodes and begins decreasing its *TP* setting and increasing *BI* due to high *LQ* values, too many neighbors discovered, and stable conditions. The source node does this as an attempt to save energy and decrease the number of neighbors discovered. As the cluster begins to move, response frames fail to reach the source node and the source node steps up its *TP* and decreases *BI*. This is depicted around the 3 second mark in Figs. 2 and 3. When the cluster moves back, toward the source node, the source node detects the change in *LQ* and again decreases *TP* while increasing *BI* to conserve energy and reduce the number of neighbors discovered. This readjustment occurs around 18 seconds, before the cluster reaches the source node location.

© 2008 ACADEMY PUBLISHER

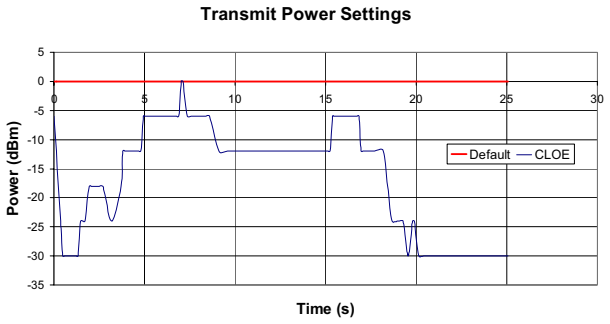

Figure 2. Transmit power settings for default and CLOE operations.

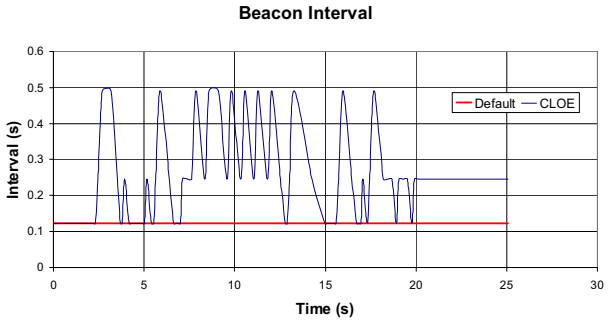

Figure 3. Beacon interval settings for default and CLOE operations.

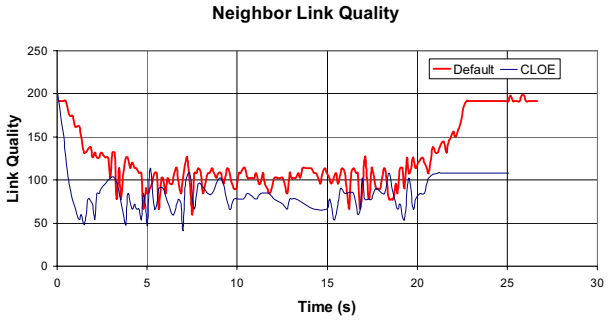

Figure 4. Link quality for default and CLOE operations.

The *TP* settings in Fig. 2 reflect the amount of energy saved by using CLOE. In this scenario, a neighbor node with CLOE uses merely 10.6% of the transmission energy needed by a default neighbor device with *TP* setting of 0 dBm. However, in situations where nodes are near the range limit, the energy savings will be significantly reduced, since CLOE will need to maximize its *TP* setting to maintain connectivity.

The *BI* values will fluctuate depending on the mobility of the neighbor node, as shown in Fig. 3. Higher values are desirable to conserve energy. The default operation was defined to transmit a beacon every 0.123 s (*BI* = 3). CLOE saves energy by increasing the *BI* setting, causing less transmissions.

The *LQ* curves generated from a neighbor with and without CLOE are shown in Fig. 4. It is interesting to note that a neighbor using CLOE generates response frames with overall lower *LQ* than a default ZigBee neighbor. This is due to the reduction of neighbor *TP* in favor of the energy saving objective. Thus, connectivity is still maintained with sufficient *TP* from both the source node and neighbor node.

2) Association Application

For the association application, we create the scenario depicted in Figs. 5 to 7. We use $n = 1$ for the association utility function to demonstrate the CLOE association process. In other words, after each discovery iteration, the source node would output the most appropriate neighbor node for association. The source node S and neighbor node A remain stationary throughout the scenario, with neighbor node B moving towards S. The transmission ranges for each node are portrayed by solid (S), dot (A), and dot-dash (B) line patterns. At time t = 0, in Fig. 5, only A is within the range of S, and B begins to approach S. At time $t = \Delta t_1$, in Fig. 6, B has arrived near A's location and they are at approximately the same distance from S. Before time Δt_1, S would only choose to associate with A. However, when B approaches A, S could associate with either A or B, because the neighbors are both within range of S and have similar settings at this point of the scenario. At time $t = \Delta t_1 + \Delta t_2$, in Fig. 7, B is closer in range to S than A. As a result CLOE reduces the *TP* for S and B such that they are maximizing their energy efficiency but are still within range of each other. Towards this point in the scenario, S would continue to choose B for association. Beacon transmissions from S would not be received by A due to range, and as a result, A would not be able to generate any responses.

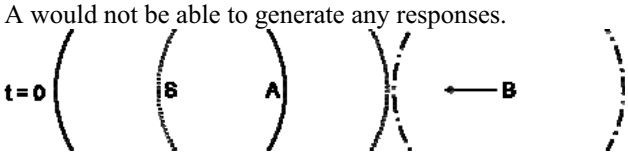

Figure 5. Association scenario at t = 0.

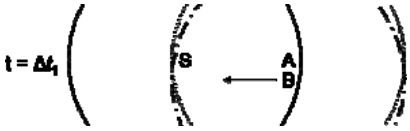

Figure 6. Association scenario at $t = \Delta t_1$.

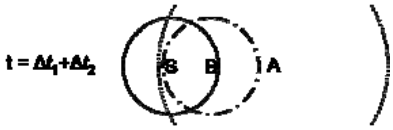

Figure 7. Association scenario at $t = \Delta t_1 + \Delta t_2$.

B. OPNET Simulation

1) Scenario 1

In the first OPNET scenario, the mobile neighbor node followed a linear trajectory, where the distance from the source node over time is shown in Fig. 8. Scenario 1 was created in OPNET to simulate the live experiment scenario.

© 2008 ACADEMY PUBLISHER

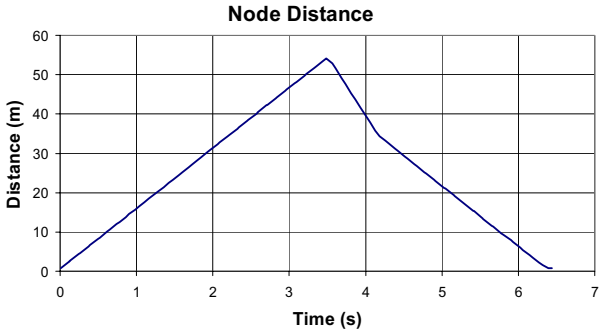

Figure 8. Distance between mobile neighbor node and source node.

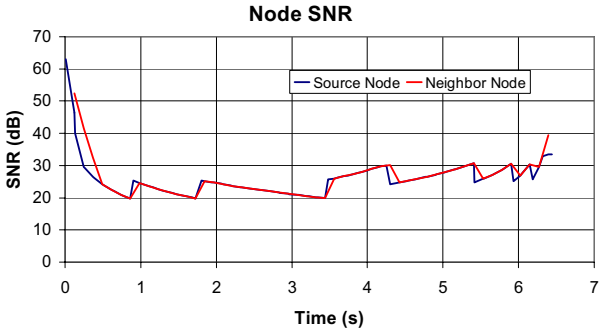

Figure 9. SNR measurements of received packets.

The source node sent beacon frames at maximum transmit power (0 dBm, or 1 mW) for ranging purposes. For data packets from both nodes, the transmit power was dynamically adjusted according to the current transmit power setting and the SNR measured at the receiving node (see Fig. 9). The raising and lowering of the transmission power setting is evident in the teeth-like graph of the source node denoted in Fig. 10. In contrast, the neighbor node generates a power curve that acts as a lower bound to the source node power curve. This is due to the fact that the neighbor node only receives beacon frames, and as a result can save significantly more energy than the source node. The energy saving feature of both nodes is more easily visualized in a moving average plot of the power curves, displayed in Fig. 11. A node using the default power setting for every transmission would have consumed 6.44 mJ of transmission energy in this scenario. Using CLOE, the source node consumed 4.13 mJ (64.14%) and the neighbor node consumed only 1.71 mJ (26.46%).

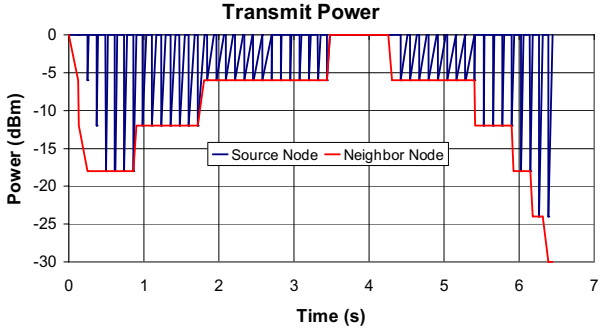

Figure 10. Transmit power settings of source and neighbor nodes.

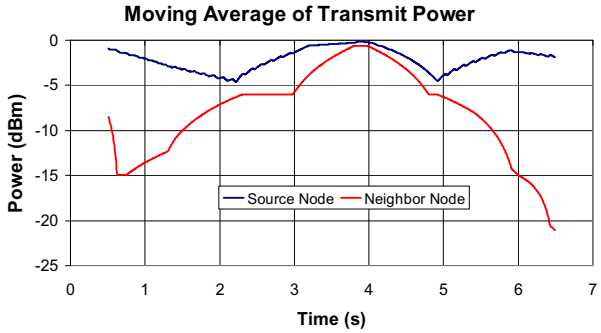

Figure 11. Moving average of transmit power settings.

2) Scenario 2

In the second OPNET scenario, four nodes were stationary (including the source node) and one was mobile. The mobile neighbor node generated a transmission power curve similar to the neighbor node curve in Fig. 10. The transmission energy consumption was also nearly identical for the mobile neighbor node in both scenarios. The stationary neighbor nodes would begin data transmissions at maximum power, then immediately reduce transmission power, if possible. For example, node 3 in Fig. 1 was approximately 50 m away from the source node (node 0) and had to transmit data at 0 dBm to create a high enough SNR at node 0. On the contrary, node 2 was located close to the source node and succeeded in dropping to a low transmit power (-18 dBm) for the duration of the simulation. The source node utilized a different transmit power setting for each neighbor node, such that overall transmission energy consumption was reduced for the network. As a result, the source node consumed slightly more energy for its transmissions and required 66.10% (4.28 mJ) of default transmission energy consumption. The source node transmit power curves are depicted in Fig. 12.

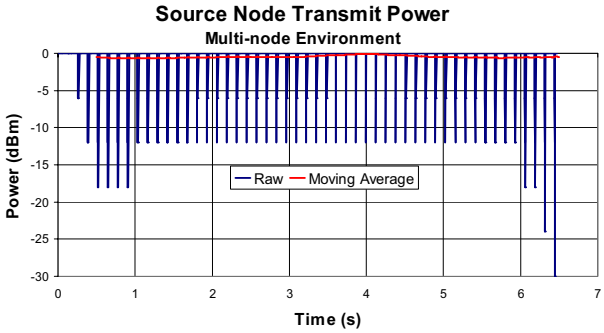

Figure 12. Source node transmit power in multi-node environment.

C. Experiment Comparison

We verified specific operations of CLOE using both live experiments and OPNET simulations. The software components and source code behind the utility functions and engine were kept the same in both environments for validation purposes. Naturally, environment-specific code varied in certain instances. For instance, all nodes were created in a single program in the simulator

© 2008 ACADEMY PUBLISHER

environment, whereas each node ran its own CLOE in the live environment.

The advantages of implementing our mechanism on a simulator are that simulation runs can be deterministic and reproducible. Seed values could be enumerated such that different runs could employ different sets of random number choices. Reproducibility also has obvious advantages when debugging the utility functions. However, certain aspects of the OPNET simulator had limited fidelity, such as simulating the physical channel. In the live scenarios, some experiment runs led to dropped frames or delayed synchronization for no apparent reason other than severe momentary fading or software interrupt resolution. These random events were not captured by the simulations. In the OPNET simulator, synchronization between nodes always occurred successfully and frames were only dropped if nodes were out of range. Thus, our simulations do not accurately account for the anomalies that occur in the live implementations, but still sufficiently represent the behavior of CLOE.

Another advantage of implementing CLOE in OPNET is that some experiments can run orders of magnitude faster than they would on a real network even after scaling the number of nodes. For the live experiments, the setup and management overhead of running the cross-layer engines in a ZigBee PAN is a significant contributing factor to longer runtimes. In addition to the typical edit-compile-debug cycle, additional steps were needed to copy the compiled binaries to all the ZigBee developer kits, start the nodes in a staggered fashion, begin logging events, and copy results from all nodes to a results repository for analysis.

Finally, implementing CLOE in the OPNET simulator was invaluable for debugging purposes but differs from reality. The differences in operations between simulated and live environments were not significant to warrant two separate behaviors of CLOE. In other words, each node with CLOE behaved appropriately independent of the environment. By capturing the strengths of both environments and being fully aware of the limitations, we were able to successfully verify operations of our cross-layer optimization engine.

VII. Conclusion

This paper describes the foundations for establishing high-level behavior using cross-layer design. A cross-layer optimization engine was designed with capabilities of measurement and performance. A feedback mechanism provided by the utility functions enables efficient cross-layer design. ZigBee technology is used in the experiments. Potential applications for our mechanism were also discussed. In this work, we primarily focus on local utility functions for cross-layer optimization of the discovery and association processes with an energy savings objective. Our results show that the utility functions can work efficiently with changes in topology. CLOE and its utility functions are also scalable due to elegant design and rule-based algorithms.

VIII. Future Work

Additional CLOE applications will be examined in future research. This includes development of global utility functions and interconnections between those functions. Global utility functions will allow improved abstract objectives of networks to work efficiently towards a goal through our mechanism of cross-layer optimization.

Inclusion of anomalies in the simulation design would improve our understanding of erratic behavior in the real world implementations of CLOE. This will also improve upon the fidelity of our simulations and feedback into our cross-layer designs.

References

[1] M. Perillo, C. Zhao, and W. Heinzelman, "On the Problem of Unbalanced Load Distribution in Wireless Sensor Networks," in *Proc. IEEE Global Telecommunications Conference Workshops*, 2004, pp. 74-79.

[2] R. Brooks, "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation,* vol. 2, no. 1, pp. 14-23, 1986.

[3] M. Smith, S. Hanemann, and B. Freisleben, "Coupled Simulation/Emulation for Cross-layer Enabled Mobile Wireless Computing," in *Proc. 2nd International Conference on Embedded Software and Systems*, 2005, pp. 375-383.

[4] D. H. Wolpert and K. Tumer, "An Introduction to Collective Intelligence," Technical Report NASA-ARC-IC-99-63, NASA Ames Research Center 1999. Available at http://xxx.lanl.gov/pdf/cs/9908014

[5] S. Lynden and O. F. Rana, "Coordinated Learning to Support Resource Management in Computational Grids," in *Proc. 2nd International Conference on Peer-to-Peer Computing*, 2002, pp. 81-89.

[6] S. Lynden and O. F. Rana, "LEAF: a Toolkit for Developing Coordinated Learning Based MAS," in *Proc. International Parallel and Distributed Processing Symposium*, 2003, pp. 135-142.

[7] B. Schilit, N. Adams, and R. Want, "Context-Aware Computing Applications," in *Proc. IEEE Workshop on Mobile Computing Systems and Applications*, Santa Cruz, CA, 1994, pp. 85-90.

[8] A. K. Dey and G. D. Abowd, "Towards a Better Understanding of Context and Context-Awareness," Technical Report GIT-GVU-99-22, College of Computing, Georgia Institute of Technology 1999. Available at ftp://ftp.cc.gatech.edu/pub/gvu/tr/1999/99-22.pdf

[9] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster, "The Anatomy of a Context Aware Application," in *Proc. 5th ACM/IEEE International Conference on Mobile Computing and Networking*, 1999, pp. 59-68.

[10] T. Selker and W. Burleson, "Context-Aware Design and Interaction in Computer Systems," *IBM Systems Journal,* vol. 39, no. 3 & 4, pp. 880-891, 2000.

[11] A. Daftari, N. Mehta, S. Bakre, and X.-H. Sun, "On Design Framework of Context Aware Embedded Systems," Monterey Workshop, Chicago, Illinois 2003. Available at http://www.cs.uic.edu/~shatz/SEES/sun.paper.pdf

[12] V. Kawadia and P. R. Kumar, "A Cautionary Perspective on Cross-layer Design," *IEEE Wireless Communications,* vol. 12, no. 1, pp. 3-11, 2005.

[13] Bluetooth Special Interest Group, "Specification of the Bluetooth System," Core Specifications, version 2.1+EDR, 2006. Available at https://www.bluetooth.org/spec/

[14] IEEE 802.11 Working Group, "Part 11: Wireless LAN Medium Access Control and Physical Layer Specifications," 1999. Available at http://standards.ieee.org/getieee802/802.11.html

[15] A. M. Flynn, R. A. Brooks, W. M. Wells, III, and D. S. Barrett, "SQUIRT: the Prototypical Mobile Robot for Autonomous Graduate Students," Technical Report AIM-1120, Massachusetts Institute of Technology, Cambridge, MA 1989. Available at https://dspace.mit.edu/handle/1721.1/6020

© 2008 ACADEMY PUBLISHER

[16] IEEE 802.15.4 Working Group, "Part 15.4: Wireless Medium Access Control and Physical Layer Specifications for Low-Rate Wireless Personal Area Networks," 2003. Available at http://standards.ieee.org/getieee802/802.15.html
[17] ZigBee Alliance, "ZigBee Specification," version 1.0, 2004. Available at http://www.zigbee.org/en/spec_download/download_request.asp
[18] J. Geier, "SNR Cutoff Recommendations," Wi-Fi Planet Tutorial, 2005. Available at http://www.wi-fiplanet.com/tutorials/article.php/3468771

Creighton Hager is working as a Senior Engineer in the MITRE Corporation. His projects are related to wireless networking and network security. He is currently researching in the area of network science and complex adaptive systems for MITRE. His prior projects include WiMAX vulnerability analysis supporting OSD/NII, NSA HAIPE Specification support, integrated Network Enhanced Telemetry research, Mobile User Objective System support, and JNO UAV waveform analysis.

Creighton received his Ph.D. degree in Electrical Engineering from Virginia Tech in 2004. He has published articles in the areas of Bluetooth vulnerabilities, energy efficient security mechanisms, context-aware adaptive security, and cross-layer optimized design. Creighton was also an entrepreneur in Manila, Philippines, and improved a private high school into a technical university. His professional memberships include IEEE and Eta Kappa Nu.

Dr. D. J. Shyy is the Principal Communications Engineer within the MITRE Corporation's Networking and Communications Department. He is also a part-time professor at Johns Hopkins University. Dr. Shyy has 18 years of industrial experiences in Telecommunications. He has significant experiences in: Bluetooth, 802.11 WLAN mesh, 802.16d/e/j/m WiMAX, CDMA2000 and W-CDMA cellular system design and optimization, and wireless communications network performance modeling and simulation.

He is a Certified WiMAX Forum RF Network Engineer. He has three US patents and one international patent in the area of wireless communications. He has more than 39 journal and conference publications. He has more than 64 contributions in IEEE 802 wireless standards meetings and WiMAX Forum. He is a voting member of 802.16 standards committee. Dr. Shyy received the M.S. and Ph.D. degrees in electrical engineering from Georgia Institute of Technology, Atlanta, GA.

Jamie Ma has 10 years of experience in network design and real time embedded software design in the telecommunication and wireless communication industry. She is a MITRE Senior Network System Engineer, focused on wireless and modeling & simulation. She has conducted WiFi, ZigBee cross-layer optimization, MPLS/BGP VPN modeling and simulation in OPNET.

© 2008 ACADEMY PUBLISHER