Cognitive Networks

Ryan W. Thomas, Luiz A. DaSilva, Allen B. MacKenzie
The Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA
email: {rwthomas, ldasilva, mackenab}@vt.edu

Abstract— **This paper presents a definition and framework for a novel type of adaptive data network: the cognitive network. In a cognitive network, the collection of elements that make up the network observes network conditions and then, using prior knowledge gained from previous interactions with the network, plans, decides and acts on this information. Cognitive networks are different from other "intelligent" communication technologies because these actions are taken with respect to the end-to-end goals of a data flow. In addition to the cognitive aspects of the network, a specification language is needed to translate the user's end-to-end goals into a form understandable by the cognitive process. The cognitive network also depends on a Software Adaptable Network that has both an external interface accessible to the cognitive network and network status sensors. These devices are used to provide control and feedback. The paper concludes by presenting a simple case study to illustrate a cognitive network and its framework.**

I. Introduction

Current data networking technology limits a network's ability to adapt, often resulting in sub-optimal performance. Limited in state, scope and response mechanisms, the network elements (consisting of nodes, protocol layers, policies and behaviors) are unable to make intelligent adaptations. Communication of network state information is stifled by the layered protocol architecture, making individual elements unaware of the network status experienced by other elements. Any response that an element may make to network stimuli can only be made inside of its limited scope. The adaptations that are performed are typically reactive, taking place after a problem has occurred. In this paper, we advance the idea of *cognitive networks*, which have the promise to remove these limitations by allowing networks to observe, act, learn and optimize their performance.

This paper is divided into four parts. Section II defines what a cognitive network is. It also examines what a successful cognitive network should be able to do, listing a few possible applications for the technology. Section III investigates the cognitive process. The paper then discusses a framework for implementing a cognitive network in section IV and investigates a case study in section V.

II. What is a Cognitive Network?

A. Background

In recent years, the words "cognitive" and "smart" have become buzzwords that are applied to many different networking and communications systems. At a minimum, in the current literature we find mention of cognitive radios [12], [8], smart radios [2], smart antennas [1], cognitive packets [5], smart packets [20] and cognitive networks [11], [18]. There does not seem to be a common, accepted definition of what these terms mean when applied to a networking technology. The common aspect in the terms mentioned above is the implication that the technology has the ability to self-modify. We argue, however, that a cognitive network goes beyond the use of self-modifying technology.

Cognitive networks have been mentioned in forward-looking papers before. Mitola makes brief mention of how his cognitive radios could interact within the system-level scope of a cognitive network [12]. Saracco refers to cognitive networks in his investigation into the future of information technology [19]. He postulates that the movement of network intelligence from controlling resources to understanding user needs will help "flatten" the network by moving network intelligence further out towards the edges of the network. Mähönen et al. discuss cognitive networks with respect to future mobile IP networks, arguing that the context sensitivity of these networks could have as interesting an application in the field as cognitive radios [11]. None of these papers, however, express exactly what a cognitive network is and how it should work.

A good model to use when examining cognitive networks is cognitive radio. There is significant research interest in this area, and although cognitive radio was well defined by Joseph Mitola [13], [12], confusion often arises when the terms Software Defined Radio (SDR) and cognitive radio are used interchangeably. An SDR is simply a radio that puts most of the Radio Frequency (RF) and Intermediate Frequency (IF) functionality, including waveform synthesis, into the digital (rather than the analog) domain, allowing for great flexibility in the modes of radio operation (called "personalities") [12]. Cognitive radio, on the other hand, sits above the SDR and is the "intelligence" that lets an SDR determine which mode of operation and parameters to use. Cognition here is used in association with a technology that operates inside a complex environment (the congested radio frequency spectrum), observes it, makes behavior choices, and receives feedback from it, all the while learning – assembling a data set that will help determine future behaviors based on past and current feedback.

Cognitive networks are likely to employ cross-layer optimizations and act simultaneously on parameters belonging to multiple layers in the protocol stack. However, cognitive networks are more than cross-layer design. Cross-layer protocols are surveyed in depth by Gong in [6]. Here, the protocols are divided into two major classes: joint-layer optimization designs and cross-layer adaptive designs. Joint-layer optimiza-

1-4244-0013-9/05/$20.00 ©2005 IEEE

tions forge together two layers and then require some sort of network information synchronization to be able to perform their global optimization algorithms. Protocols in the cross-layer adaptive classification communicate information between two distinct network layers, which then adapt parameters in an attempt to locally optimize performance. Regardless of which class of cross-layer networking protocol is implemented, these solutions typically focus on a single parameter to optimize on. Furthermore, many cross-layer designs are so focused on optimizing at the chosen layers that they may end up degrading the overall system or connection performance [10]. In this manner, cross-layer design would be more aptly associated with a "cognitive layer" technology than a cognitive networking technology, since cross-layer design does not exhibit the breadth of options that should be available to a intelligent network, nor the end-to-end scope necessary to be termed a cognitive network.

B. Requirements

A cognitive network should provide, over an extended period of time, better end-to-end performance than a non-cognitive network. Cognition could be used to improve resource management, Quality of Service (QoS), security, access control, or many other network goals. Cognitive networks are only limited by the adaptability of the underlying network elements and the flexibility of the cognitive framework. In this manner, cognitive networks are not limited to only wireless networks. Ad-hoc networks, infrastructure-mode wireless networks, fully wired networks and heterogeneous networks are also candidates for cognitive network design.

Cognitive networks should use network metrics and patterns as input to the decision making process and then provide output in the form of a set of actions that can implemented in modifiable network elements. Ideally, a cognitive network should be forward-looking, rather than reactive, and attempt to adjust to problems before they occur. Finally, the architecture of a cognitive network should be extensible and flexible, supporting future improvements and elements.

Implementing a cognitive network requires a system that is more complex in terms of overhead, architecture and operation. For cognitive networks to be justifiable, the performance improvement must outweigh these additional complexities. For certain environments, such as static wired networks with predictable behavior, it may not make sense to convert to cognitive behaviors. Other environments, such as heterogeneous wireless networks, may be ideal candidates for cognition.

C. Definition

We suggest the following definition of a cognitive network:

> A cognitive network has a cognitive process that can perceive current network conditions, and then plan, decide and act on those conditions. The network can learn from these adaptations and use them to make future decisions, all while taking into account end-to-end goals.

This definition mimics standard models of cognition and learning. The network and end-to-end aspects of the definition, however, are critical to differentiating a cognitive network from other cognitive communication technologies. Without this scope, the system is perhaps a cognitive radio or layer, but not a cognitive network. Here, end-to-end denotes all the network elements involved in the transmission of a data flow. For a unicast transmission, this might include the subnets, routers, switches, virtual connections, encryption schemes, mediums, interfaces, or waveforms, to mention just a few. The end-to-end goals are what gives a cognitive network its network-wide scope, separating it from other technologies, which have only a local, single element scope.

In order to fit this definition, a cognitive network must have elements in a Software Adaptable Network (SAN) to modify. Similar to a cognitive radio, which depends on an SDR to modify aspects of radio operation (e.g. time, frequency, bandwidth, code, spatiality, waveform), a SAN depends on a network that has one or more tunable elements. Practically, this means that a network may be able to modify one or several layers of the network stack in its member nodes. A simple example of a SAN could be a wireless network with directional antennas (antennas with the ability to scan their receive or transmit strength to various points of rotation). This network would meet the definition for basic functionality of a SAN, since it contains a modifiable medium access scheme. However, it could only be called a cognitive network if the behavior modifying the antenna direction is cognisant of how those choices affect end-to-end goals. If it is only aware of how it affects link-level goals, it is simply a smart antenna or less.

D. Examples and Applications

Examples of potential cognitive network applications include:

Heterogeneity. For networks that employ a wide variety of protocols and physical layer interfaces, a cognitive network can provide a mechanism for creating order in chaos. Since a cognitive network views and learns from the observed network status, it can de-conflict individual nodes and optimize the connections – from top level objectives such as creating efficient homogeneous clusters to lower-level goals such as reducing the total amount of energy expended.

QoS. In a more general sense, cognitive networks can be used to manage the QoS for a connection. Utilizing the feedback about observed network conditions, the cognitive network can identify bottlenecks, estimate guarantees, change prioritization and optimize behaviors to provide the desired end-to-end QoS.

Security. Cognitive networks could also be used for security purposes such as access control, tunneling, trust management or intrusion detection. By analyzing feedback from the various layers of the network, a cognitive network can find patterns and risks and then react by changing such security mechanisms as rule sets, protocols, encryption and group membership. Such cognition may also aid in trust and reputation mechanisms.

III. The Cognitive Process

The central mechanism of the cognitive network is the cognitive process. This is the process that does the actual learning and decides on the appropriate response to observed network behavior. How the cognitive process operates depends heavily on whether it is implemented in a centralized or decentralized fashion and on the amount of network state known to the process.

A. Scope

As defined in section II-C, a cognitive network operates in light of end-to-end goals. This means that the scope of the cognitive network is operating above the goals of the individual network elements. Instead, it operates within the scope of a data flow, which may include many network elements. Many flows may traverse a single network element, which means that the cognitive network needs to be able to prioritize these flows. By interacting with the SAN, the cognitive network tries to maintain a set of end-to-end goals (such as routing optimizations, connectivity, trust management, etc.) by modifying the elements of the SAN. The cognitive elements associated with each flow are allowed to act selfishly and independently (in the context of the entire network) to achieve local goals.

For some network applications however, the individual elements may not be so selfish or independent. For instance, in homogeneous user environments such as military, law enforcement or corporate networks, the nodes are as concerned with the overall goals of the network as they are with their own goals. This raises the scope of a flow beyond its own requirements and is in contrast to a commercial ISP WAN (or some other network made up of unrelated users) in which the nodes are likely to behave in selfish, local-optimization modes of operation. In attempting to accommodate network-wide goals, however, more state information will need to be communicated, leading to higher overhead and complexity.

An interesting question is how much of a performance difference there is between these two scopes of operation. The performance difference between following selfish, local goals and the acting in a communal, network-wide mode of operation is called *the price of anarchy* in [17]. This term is defined to be the difference in performance between a network run by an all-knowing benevolent dictator (that can specify the "correct solution" for the connections to be at their optimum at any given time) and one governed purely by selfish anarchy. The answer to this question will guide the development of cognitive networks by indicating the required complexity and amount of state information needed to meet the goals of the cognitive network. If it turns out that the performance is significantly poorer when acting selfishly than acting communally, the cognitive network may need to provide more centralized guidance or an appropriate incentive structure to the network elements than simply a end-to-end goal. It is possible that the answer to this question will depend on the kinds of optimizations the cognitive network is trying to make.

B. State

The effect of a cognitive network's decisions on the network performance depends on the amount of network state information available to it. In order for a cognitive network to make a decision based on end-to-end goals, the system must have knowledge of the current state and the current network goals. If a cognitive network has knowledge of the entire network's state, cognitive decisions should be more "correct" than those made in ignorance. For a large, complicated system such a computer network, it is unlikely that the cognitive network would know the total system state. There is often a very high cost to communicate this information beyond those network elements requiring it, meaning a cognitive network will have to work with less than a full picture of the network status.

There are two possible modes of operation for the cognitive network and the state distribution. The first is a centralized mode of operation, in which a central server or entity tracks all end-to-end goals and network status data. The other mode is distributed, in which nodes in the network share this information and no central authority exists to manage the gathering and dissemination of this data. The trade-offs between each of these modes of operation have been examined in other contexts and the usual problems associated with them are valid here too. From the possibility of a single-point of failure problem in the centralized mode to overhead issues in the distributed mode, system designers will need to determine which mode offers the best trade off for their cognitive design. Beyond purely centralized or distributed, hybrid mechanisms may also be possible. An example of this could be a network that transmits end-to-end goals in a distributed fashion but uses a central repository to store network status updates.

Additional problems that a cognitive network will have to solve include: Where does this network state information come from? How do the network elements distribute the network state information amongst each other? If the network state is reported by the members of the network, how do we enforce truthful reporting?

C. The Feedback Loop

A unifying aspect of any learning model is the feedback loop. In order to learn, a feedback loop is created in which past interactions with the environment guide current and future interactions. Figure 1 illustrates a simple example of a feedback loop first put forward by Col John Boyd, USAF (ret). Commonly called the "OODA" loop [3], standing for Observe, Orient, Decide and Act, the model was originally used to help military officers understand the thought processes of their adversaries. However, this loop has been adopted outside of the military, in applications ranging from business management to artificial intelligence. It is remarkably similar to the model Mitola uses to describe the cognition process in cognitive radios. The loop consists of four self-explanatory components, which guide a decision maker through the process of choosing an appropriate action based on input from the environment. The loop is missing a few important components, however. One is an overarching goal, which should feed

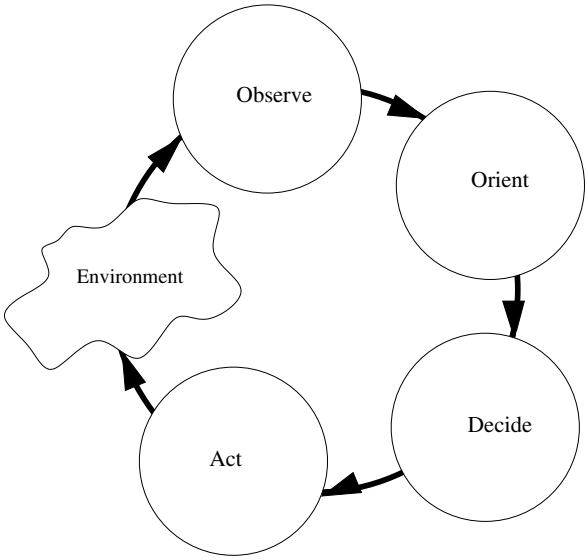

Fig. 1. The OODA Loop

in from outside the loop and guide the "orientation" and "decision" components by providing a context in which to make a decision. Another missing component is a learning module, which prevents mistakes from previous iterations from being made on future iterations.

Feedback loops such as the OODA loop work because although the environment in which the decisions are being made may be highly complex, it is not totally random. There is a structure to the complex system that may not be apparent from outside analysis but an attempt to approximate it can be made through iterative cycles of a test-response feedback loop. By simplifying the environment to a black box model, it may be possible to determine some of this structure, particularly if the system is reasonably stationary within the time frame of interest for the adaptations. In a cognitive network, network elements and their interactions are the "black box." The SAN modifications that the cognitive network implements act as inputs to the box, while behavior observed in the network are the outputs.

The problem with simple cross-layer protocols (such as the "cognitive layers" mentioned before) is that they do not model the entire end-to-end connection as the black box, but instead they examine a specific subset of the network. This can produce unforeseen responses at other elements of the network that are not a part of the cross-layer protocol. These responses are outside of the adaptive mechanism or the optimization algorithm used by the cognitive layer. Ideally, a cognitive network would be able to optimize the connection by changing many of the flexible options available to it in the network. Assuming that there is some underlying structure to the response of the network, some form of machine learning algorithm could be used to analyze the network response and choose the correct set of inputs to optimize the system.

D. Cognition

Cognitive processes fall under the title of "machine learning" and have been an active research area since the 1960s. Machine learning is broadly defined in [21] as any algorithm that "improves its performance through experience gained over a period of time without complete information about the environment in which it operates." Underneath this definition, many different kinds of artificial intelligence, decision making and adaptive algorithms can be placed, giving cognitive networks a wide scope of possible mechanisms to use in learning.

In the following paragraphs, a brief discussion of possible mechanisms for machine learning is given. There are many other possibilities for implementing machine learning, including combinations or hybridization of the following strategies; the choice of machine learning algorithm depends on what the the network goals are and how these problems are set up. Complex cognitive networks may have several cognition processes operating, each using mechanisms appropriate for the problem being solved.

When discussing machine learning and artificial intelligence, the area of neural networks is often cited as a successful mechanism for implementing learning. Neural networks use a bottom-up method of learning, simulating the biological neurons and pathways that the brain is thought to use. A series of these artificial neurons analyze different aspects of known inputs with some amount of unknown corruption. Pattern recognition is a common and straightforward application of cognitive networks. If network responses are modeled as a noisy pattern, a neural network could be used to categorize the pattern into predetermined responses.

Genetic algorithms are usually used in optimizing over very large solution spaces where exhaustively searching would be too costly. By imitating the process of evolution (selection, recombination and mutation), genetic algorithms are able to explore these large solution spaces for local optima. Genetic algorithms have many applications, but work best for centralized problems where the environment is well known. For this reason, if most of the current network state is known, genetic algorithms could be used to determine optimal behaviors.

Recently, artificial intelligence has focused on expert systems [16]. Expert systems are often used to make decisions in a very narrow field of knowledge (a common application is medical diagnosis) and imitate the "intuition" that a professional in the field would have in finding a solution by asking a series of questions (typically with only "yes," "no," and "no response" answers). Expert systems can be useful in determining how to prioritize goals and solve problems that have a limited number of possible inputs.

Kalman filters [4] are not usually considered a machine learning algorithm, but do fit the definition of machine learning given above. They contain an adaptive algorithm for feedback control. Usually found in systems that contain significant amounts of system noise, the Kalman filter is a recursive filter used to estimate the actual and future state of the system based on noisy Gaussian measurements. The ability to act as a

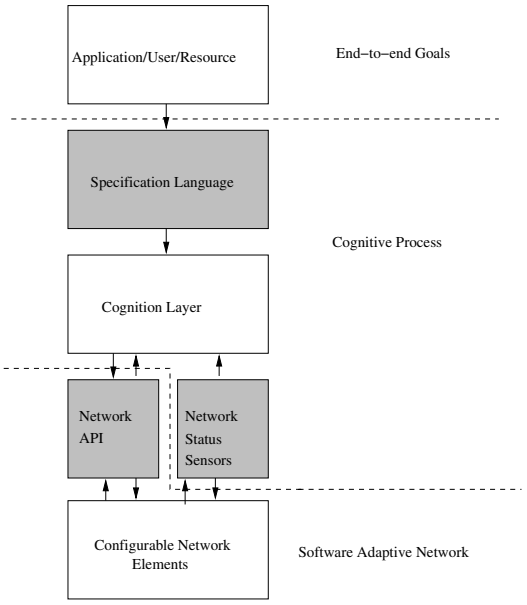

Fig. 2. The Cognitive Framework

dynamic filter means a Kalman filter can be useful for tracking and maintaining a particular performance metric in a changing or noisy system.

Learning automata [15] are a simple method of intelligently learning a process to an unknown feedback system. Typical applications for learning automata include problems where there are many dynamic elements interacting with a complex system, such as intelligent vehicles [22], cross-layer optimization problems [7] or routing problems [15]. If the problem is distributed and requires very little state information, learning automata can be a good approach.

Regardless of what implementation is chosen, the process needs to be able to learn or converge to a solution faster than the network status changes, and then re-learn and re-converge to new solutions when the status changes again. The issue of convergence is of particular importance in environments that change frequently, such as mobile wireless networks.

IV. A Cognitive Network Framework

Regardless of what the scope of a cognitive network is, how decentralized it is or how much state information is available, some aspects of the network have to be known. A network node makes use of a communication medium and as such requires at least two entities synchronized in certain operating modes to communicate effectively. Furthermore, the cognitive network has to know what its goals are and how to interact with the underlying SAN. This information requires a software framework to be in place, tying the user needs, cognition and underlying network together. Figure 2 illustrates the nominal architecture put forth in this section to build the cognitive network.

A. User / Application / Resource Requirements

The top level component of the cognitive network includes the end-to-end goals, which are put forth by the network users, applications or resources. These requirements drive the cognitive behaviors by identifying, prioritizing and weighting the requirements to the user of the cognitive network.

As mentioned in section II-C, these end-to-end goals are what sets a cognitive network apart from other cognitive communication mechanisms. Changing or modifying a network element can create performance optimization local to the particular element being modified, but still cause a negative effect on the performance elsewhere in the network or node. For instance, a particular wireless MAC protocol may optimize for power consumption, creating higher hop count routes that use short links. However, this mode of operation might result in additional end-to-end delay (due to the additional processing, queueing and transmission delay that goes along with higher hop count routes) which in turn could affect the transport layer, leading to more retransmissions. As a result, the overall power consumption for the node might be higher than without the original optimization. Alternatively, if the protocol does achieve lower power consumption, it may increase the jitter of the connection beyond what is feasible for a voice application. The possibilities of interaction are endless. Only if the users of the network reveal their requirements can a cognitive network know if it is operating "successfully."

Like most engineering problems, there is likely to be an engineering trade-off for every goal optimized on. In the optimization space, the cognitive network will not be able to optimize all metrics indefinitely. In most systems, a point is reached in which one metric cannot be optimized without affecting another. At this point, the cognitive network will need to know the metric priorities and the minimum and maximum required performance of each of these metrics.

B. Cognitive Process

The cognitive process consists of three components: the specification language, cognition layer, and network input. These components follow the feedback loop discussed in section III-C, providing the actual intelligence of the cognitive layer, and allow it to interface with the SAN and the network users.

Cognitive Specification Language: To connect the requirements of the top-level users of the network to the cognitive layer, an interface layer must be developed. This information need not be globally known (unless the cognitive network is operating with a system-wide scope, in which case global knowledge might be required), but must be communicated between the source of the requirements and the local cognitive process.

This process is similar to the Radio Knowledge Representation Language (RKRL) proposed by Mitola for cognitive radio [13] or QoS specification languages. There are already several different QoS specification paradigms in existence [9] and the concept of these languages – mapping requirements to underlying mechanisms – is the same here, except that

the mechanisms are adaptive to the network capabilities as opposed to a fixed set of mechanisms.

Unlike RKRL or a QoS specification language, a Cognitive Specification Language (CSL) must be able to adapt to new network elements, applications and goals, some of which may not even be imagined yet. For this reason, an extensible format such as eXtensible Markup Language (XML) may be appropriate. Other requirements may include support for distributed or centralized operation including the sharing of data between multiple cognition layers. This language should not actually perform the cognitive process (this is done by the cognition layer) but an application may be needed to translate the requirements (as generated by the top level users) into CSL.

Cognition Layer: The cognitive aspect of the network could be either centralized or distributed. Depending on whether the network is operating in local or communal mode, distributed or centralized may be a more natural fit. It is a reasonable assumption, however, that most networks will have the cognitive behavior handled at each node, providing an argument to decentralize the operation. Possible algorithms for this layer were discussed in section III-D.

Network Status Sensors: While the CSL provides end-to-end goals as input to the cognition layer, the network status sensors provide feedback from the network to the cognition layer. They also allow the cognition layer to observe patterns, trends, and thresholds in the network for possible action. The network status sensors may only report data from the connection that the cognitive network is managing, or they may distribute their information to the entire network. As discussed in section III-B, the status these sensors collect can be communicated in either a distributed or centralized form. Such sensors can be co-located with (or be a specific function of) general purpose network nodes.

C. Software Adaptable Network

The SAN is really a separate research area, just as the design of the SDR is separate from the development of the cognitive radio. However, the cognitive network needs to be aware of the interface that the SAN provides to the network elements it can control. This is similar to an Application Programming Interface (API) or an Interface Description Language (IDL). The SAN also consists of Modifiable Network Elements, which act as points of control for the cognitive network.

Network API: The actual software for the SAN and its interface will likely sit as some sort of middleware between the user or application and the network elements, including the network stack. Using multi-platform glue such as Common Object Request Broker Architecture (CORBA), the API will bridge this gap. Just like the other aspects of the framework, the API should be flexible and extensible. Continuing our analogy with SDRs, an existing system that is possibly analogous to this is the Software Communications Architecture (SCA) used in the Joint Tactical Radio System (JTRS) .

Another responsibility of the API (and the SAN) is to notify the cognitive network of what the operating states of the network elements are. Many modifications to the network stack require that both ends of the link be synchronized and operating in the same mode. The communication required to synchronize these states is the responsibility of the SAN and could be accomplished either in or out of channel. Either method is a significant hurdle to overcome. If, for instance, two cognitive network processes switch to different transmission frequencies, or different byte orders in the packet headers, or even incompatible retransmission strategies, communication between the two nodes could break down.

Thus at a minimum, the cognitive network needs to know what modifications have been made by each of the devices in its communication channel that could impede communication if not synchronized. Since it is possible that different networks may have different modifications available to them, the system used to gather and distribute this information must be robust and extensible.

Modifiable Network Elements: The actual components of the SAN are the modifiable network elements. These elements can include any object or element used in a network, although it is unlikely that all elements in a SAN would be modifiable. Each element should have public and private interfaces to the API, allowing it to be manipulated by the SAN and the cognitive network.

V. Case study

In order to illustrate some of the concepts discussed in this paper, we have developed a simple case study. This study consists of a distributed cognition layer made up of learning automata. Each node in the network is equipped with a directional reception antenna and omnidirectional transmission antenna. The omnidirectional antenna has a variable power source that can vary the amount of power transmitted. The end-to-end goal is to maximize the connection time for unicast and multicast communication between a source node and one or more destination nodes. In section V-A, this problem will be mapped onto the cognitive network architecture discussed previously. In section V-B, the simulation used to model this problem is discussed. Finally, results and conclusions from this simulation are presented in section V-C.

A. Architecture

User/Application/Resource Layer: For many applications, such as voice and video, having a long-lasting end-to-end connection is an important consideration. The stability and lifetime of a connection is a limiting factor. Many factors may affect the expected lifetime of a network connection in a wireless network. For instance, traffic congestion can cause timeouts in upper layer protocols, interference can cause loss of connection at the physical layer, and mobility can cause unexpected disconnections in traffic routing. However, for mobile and portable devices, one of the biggest factors in the lifetime of a connection is the utilization of the available power capacity contained in the batteries of the network devices.

For this study, it is assumed that the wireless network is made up of a collection of network elements with varying

power capacity limitations. Some elements may be battery powered, with limited power capacity, while others may be less mobile, wired into the electrical system with almost limitless power. The lifetime of a data flow, however, is limited by the rate at which the energy is being consumed and the power capacity of the nodes in the network. By minimizing the total utilization of the flow, the lifetime of devices being used to carry the connection can be maximized.

Furthermore, for this particular case study, there are no additional requirements or limitations set by the top layer. If another goal was to be set, such as a "fair" utilization of all nodes in the network or minimizing the power consumption of the path, a prioritization scheme would have to be developed to accomplish this multi-objective optimization. To assist in prioritizing the utility of different combinations, performance bounds may need to be set on each of the goals.

Cognitive Specification Language: The Cognitive Specification Language translates the end-to-end goals to a form understandable by the cognitive process. For the goal of maximizing flow lifetime, the specification layer translates these goals in terms of the modifiable network elements. In this case, it is relates the goal of connection lifetime to the total power utilization of the chosen route. More complex cognitive specification layers could also relate this goal to other possible factors, such as traffic rate, signal-to-noise ratio, or mobility. This would depend on the cognition layer's ability to process these additional goals and having modifiable network elements that are able to support them. If user mobility is not something that can be controlled by the software adaptable network, the cognitive network cannot specify a goal to the cognition layer that requires mobility to be modified. Similarly, if the cognition layer is not capable of optimizing multiple goals, specifying objectives for both power utilization and signal-to-noise ratio would not be possible.

In this simple case, the network flow lifetime goal is specified to the cognition layer by a fitness function that describes the total power utilization. With the goal of minimizing the power utilization of the multicast route, the fitness function returns higher values for power combinations that lead to lower utilization. We define a multicast route as a tree T. The tree is anchored at a source node s that transmits to destination nodes D (where $|D|$ can range from 1 to $|N|-1$). We assign a cost function to each edge of the tree that is the utilization ratio of the power expended to power capacity of the transmitting (source) node of that edge:

$$u(t) = \frac{p(t)}{c(t)} \tag{1}$$

Here $p(t)$ is the power set for transmission from node t and $c(t)$ is the available power capacity for node t. $c(t)$ is a function of node t's battery capacity in coulombs, the average duty cycle of a transmission, and the potential drop across the battery. We then additionally define maximum and minimum costs for an edge as

$$u_{max}(t) = \frac{p_{max}(t)}{c(t)} \tag{2}$$

$$u_{min}(t) = \frac{p_{min}(t)}{c(t)} \tag{3}$$

where $p_{max}(t)$ is an upper limit on the power transmitted from node t and $p_{min}(t)$ is a lower limit on power transmitted from t. For a given multicast tree T the fitness function $\beta \rightarrow [0,1]$ maps the route's power utilization to the end-to-end goal of maximizing connection lifetime by defining β as

$$\beta = \begin{cases} 0 & D \nsubseteq T \\ 1 - \frac{\sum_{t \in T} u(t) - \sum_{t \in T} u_{min}(t)}{\sum_{t \in T} u_{max}(t) - \sum_{t \in T} u_{min}(t)} & D \subseteq T \end{cases} \tag{4}$$

The fitness function is 0 if the tree is broken due to interference or having inadequate power to reach all all branches. The function then increases as the average utilization decreases, until it is maximized with a value of 1 in the case that the link cost is u_{min} for each node in the tree.

Cognition Layer : In [23], the heuristic approach to directional antenna multicast, Directional Reception Incremental Protocol (DRIP) was introduced. It was shown to be remarkably close to the optimal min-power solution. However, DRIP is designed to work in the absence of inter-node interference. Wood and DaSilva identified that heuristic solutions have good performance in optimal environments, but have trouble when the real-world problem of interference is introduced. The heuristic works by assigning a cost to each link related to the interference-free power required to reach a particular node. However, once the tree is built, interference will creep into the system and require more power than initially calculated. This problem can be somewhat mitigated by estimating the interference and augmenting the required link power. However, this estimate is difficult to accurately calculate since the actual amount of in-system interference each node receives is a function of the power output of neighboring nodes, the distance to the neighbors, and whether the neighbors reside in the directional beam of the receiving node. This case study uses a modified DRIP algorithm that attempts to make a best estimate of the required power making an estimate of the expected amount of in-system interference. However, if this estimate is too high, more power (and battery utilization) will be used than necessary, and if it is too low, connections (and the multicast tree) will be broken because of interference.

The cognition process is the layer that solves the mis-estimation problem and simultaneously minimizes the power used. By interacting with the environment, it "learns" to optimize the network by adapting the power output of all interior tree nodes to a lower power utilization ratio while maintaining a fully connected multicast tree. To solve the connection lifetime problem, the cognition layer is implemented as a network of learning automata. An automaton is a learning model that has an action space (either discrete or continuous), a probability vector that maps a selection probability to every action, and a feedback function. Each automaton in this

example is created as an Finite Action Learning Automata (FALA). Specifically, the automata are deigned as Learning Reward-Inaction automata (denoted $L_{\text{R-I}}$). $L_{\text{R-I}}$ automata and their behavior are discussed in detail in [14], [15], [21]. An $L_{\text{R-I}}$ automaton is so named because the probability vector of the automaton is only reinforced when the automaton performs an action that receives positive feedback. If the feedback is 0, no updating occurs.

As mentioned before, an $L_{\text{R-I}}$ automaton works by updating the probability that it will select actions in the future based on feedback from current and previous actions. In this case, each automata in the tree is given 10 possible power levels to choose from. These actions are represented in elements of the action vector $\hat{\alpha}$. The probability of choosing any particular action in $\hat{\alpha}$ is contained in the probability vector $\hat{p}$. These two vectors are related to one another through their indices. Initially, the probability vector is uniformly distributed, meaning that each element of $\hat{p}$ is set to $|\hat{p}|^{-1}$. The probabilities change as the automata get feedback from the system. If the action chosen at instance k is α_i and it receives feedback $\beta(k)$, the probability vector $\hat{p}$ is updated in the next time increment as:

$$p_j(k+1) = \begin{cases} p_i(k) + \lambda\beta(k)(1 - p_i(k)) & j = i \\ p_j(k) - \lambda\beta(k)(p_j(k)) & \forall\, j \neq i \end{cases} \quad (5)$$

where λ is the learning parameter and i denotes the action choice selected. Equation 5 shows that the probability of choosing a particular action increases when $\beta > 1$ but does not change when $\beta = 0$. It can be easily shown through induction that $0 \leq p_j(k) \leq 1$ and $\sum_{j=1}^{|\hat{p}|} p_j(k) = 1$ for every value of k. The automaton calculates β using the function defined by the Cognitive Specification Language.

When the FALA loop through the choice-action-feedback cycle, they generate a Markov process in the probability vector $\hat{p}$. The state space for this process can be given by

$$S_r = \left\{ \hat{p} | \hat{p} = [p_1, p_2, \ldots, p_r]^T; 0 \leq p_i \leq 1 \,\forall i, \sum_{i=1}^{|\hat{p}|} p_i = 1 \right\} \quad (6)$$

The interior of this state space, S_r^0 is the set of all possible valid $\hat{p}$ in S_r, and is the search space of the FALA. This is because when a FALA operates, it is trying to solve the optimization problem

$$\max_{\hat{p}} E[\beta(k) | \hat{p}(k) = \hat{p}] \quad (7)$$

which, assuming small enough learning parameter λ, is at a local maxima when $\hat{p}$ is a unit vector with one component of value one and all others of value 0.

In this example, the automata learn by repeatedly selecting power levels to transmit at from the action space and the determining the feedback from the β function provided to it by the cognitive specification language. The process is repeated until the system converges and stabilizes to a local utilization minima.

Network API: The network API allows the cognition layer to interface with the network. For this study, the cognition layer needs to control the output power from the transmitting nodes. The API should provide the cognition limits on the possible power settings, and provide a hook for setting the power output requested by the cognition process. Additionally, the API should provide a mechanism for the cognition layer to share information between nodes. In order to calculate the fitness function, members of the multicast tree have to share their utilization ratios with each other. Since child nodes have to communicate with parent nodes as well as parent nodes communicating with child nodes, the API should provide some basic form of bi-directional communication mechanism that goes beyond the directional multicast discussed here.

Network Status Sensors: Network status sensor read information from the network and provide it to the cognition layer. For the case study, the status sensors are very simple, measuring the power output and capacity at each node, as well as the connectivity of the network.

Modifiable Network Elements: The only modifiable network element for this study is the power output of the transmitter. Although other elements, such as the beam direction, may be modifiable, they are not controllable by the cognitive network.

B. Simulation

The network was simulated using Matlab. A 10 unit by 10 unit map was created, and 10 ad-hoc nodes were placed in the map using a uniform distribution for the x and y coordinates. The receive beamwidth was set to 45 degrees wide, and we assumed that 70% of the power was received in the main lobe. These nodes are each given one $L_{\text{R-I}}$ automata to control the transmission power, varying it among 10 power levels uniformly distributed from 10% to 110% of the power estimate provided by the DRIP heuristic. Unless otherwise specified, all results refer to a learning parameter λ of 0.10.

The simulation models the heterogeneity of the battery capacity by assigning random battery capacities to each node. There is a 30% chance that a node will be powered by a generated power source instead of a battery. The power capacities are uniformly distributed from 300 to 1000 units of power. The signal-to-noise ratio threshold for communication to occur was set to a value of 1, meaning that signals received with a power level greater than the noise and interference will be correctly received. Out-of-system noise was set to 0.1 units. It was experimentally determined that setting the in-system interference estimate to 0.45 units of power gave the highest probability of successfully connecting the tree.

The number of receivers was varied from 1 to 9, with 100 runs performed for each receiver set size. For each simulation run, the average utilization ratio for the tree was measured, as well as the single-node maximum utilization, the fitness function value and the iterations to convergence. A simulation was determined to have converged when all vectors $\hat{p}$ had an element greater than $1 - \varepsilon$. For this simulation ε was set to 10^{-4}.

It should be noted that this simple case study simulation ignores unknown secondary effects that could work against the network goals. A more complex example might have more network status sensors. This would allow other variables

		Number of Receivers				
		1	3	5	7	9
	0.05	74%	74%	72%	71%	71%
λ	0.10	70%	69%	69%	66%	70%
	0.15	71%	66%	66%	66%	64%

TABLE I

MEAN PERCENT IMPROVEMENT IN POWER UTILIZATION (100 TRIALS)

that effect the connection lifetime to be determined, possibly changing either the function β provided by the CSL or the value that β gives the cognition process.

C. Results

In order to determine the optimality of the cognitive network solution, the cognitive network performance is compared against the non-cognitive heuristic DRIP solution.

Because of its reliance on estimating the out-of-system noise, the heuristic solution does not always determine a workable solution. Across all receiver sets, the heuristic finds a working solution only 86% of the time. In contrast, the cognitive network finds a solution 97% of the time, an 11% improvement.

In terms of average power utilization, the cognitive network out-performs the heuristic. Removing the cases where the heuristic does not find a solution and the where the source node has an infinitely large power source (the solution to this is trivial and both methods achieve average utilization of 0), the cognitive network performs at least 64% better than the heuristic. Table I shows the average improvement for the cognitive network over the non-cognitive network.

Since the multicast tree is like a multi-ended chain, should the connection fail between any two links in the chain, the tree will fail to reach all recipients. The lifetime of the tree is proportional to the most utilized link, since it will fail first. Although the cognition layer does not specifically try and determine the routing system containing the node with the minimal maximum utilization, the cognitive network does have an advantage over the heuristic in this aspect also. In fact, the single node maximum utilization ratio of the cognitive network is, on average, 70% lower than that of the heuristic.

VI. CONCLUSION

This paper has presented, defined and described a new form of adaptive network technology, the cognitive network. Although the term "cognitive" is associated with many communications technologies, we have defined a cognitive network to be a unique, adaptive networking technology that operates in light of the end-to-end goals of a data flow. This paper described the software framework required to implement a cognitive network, and introduced the idea of a SAN, an underlying network technology that has modifiable network elements. By presenting a case study, this paper illustrated both the architecture and potential of a cognitive network. Cognitive networks are a future networking technology that will allow data networks operating in complex, heterogeneous, noisy and dynamic environments to reclaim stability by learning and adapting their behaviors to meet top-level end-to-end goals.

REFERENCES

[1] Angeliki Alexiou and Martin Haardt. Smart antenna technologies for future wireless systems: trends and challenges. *IEEE Communications Magazine*, 42(9):90–97, 2004.

[2] Robert Berezdivin, Robert Breinig, and Randy Topp. Next-generation wireless communications concepts and technologies. *IEEE Communications Magazine*, 40(3):108–116, 2002.

[3] John Boyd. A discourse on winning and losing: Patterns of conflict. 1986.

[4] Eli Brookner. *Tracking and Kalman Filtering Made Easy*. Wiley-Interscience, 1998.

[5] Erol Gelenbe, Ricardo Lent, and Zhiguang Xu. Design and performance of cognitive packet networks. *Performance Evaluation*, 46(2-3):155–176, 2001.

[6] Michelle Gong. *Improving the Capacity in Wireless Ad Hoc Networks through Multiple Channel Operation: Design Principles and Protocols*. PhD thesis, Virginia Polyechnic Institute and State University, 2005.

[7] M. A. Haleem and R. Chandramouli. Adaptive downlink scheduling and rate selection: A cross-layer design. *IEEE Journal on Selected Areas in Communications*, 23(6):1287–1297, 2005.

[8] Simon Haykin. Cognitive radio: Brain-empowered wireless communication. *IEEE Journal on Selected Areas in Communication*, 23(2):201–220, February 2005.

[9] Jingwen Jin and K. Nahrstedt. QoS specification languages for distributed multimedia applications: A survey and taxonomy. *IEEE Multimedia*, 11(3):74–87, 2004.

[10] Vikas Kawadia and P. R. Kumar. A cautionary perspective on cross-layer design. *IEEE Wireless Communications*, 12(1):3–11, 2005.

[11] Petri Mähönen, Janne Riihijärvi, Marina Petrova, and Zach Shelby. Hop-by-hop toward future mobile broadband IP. *IEEE Communications Magazine*, 42(3):138–146, 2004.

[12] Joseph Mitola. *Cognitive Radio: An Integrated Agent Architecture for Software Defined Radio*. PhD thesis, Royal Institute of Technology (KTH), 2000.

[13] Joseph Mitola and G. Q. Maguire. Cognitive radio: making software radios more personal. *IEEE Personal Communications*, 6(4):13–18, 1999.

[14] Kaddour Najim and Alexander S. Poznyak. *Learning Automata: Theory and Applications*. Pergamon, 1994.

[15] Kumpati S. Narendra and Mandayam A.L. Thathachar. *Learning Automata: An Introduction*. Prentice Hall, 1989.

[16] Nils J. Nilsson. *Artificial Intelligence: A New Synthesis*. Morgan Kauffman Publishers, 1998.

[17] Christos H. Papadimitriou. Algorithms, games, and the Internet. In *Proceedings of STOC 2001*, 2001.

[18] Chris Ramming. Cognitive networks. In *DARPATech*, 2004.

[19] Roberto Saracco. Forecasting the future of information technology: How to make research investment more cost-effective. *IEEE Communications Magazine*, 41(12):38–45, December 2003.

[20] Beverly Schwartz, Alden W. Jackson, W. Timothy Strayer, Wenyi Zhou, R. Dennis Rockwell, and Craig Partridge. Smart packets for active networks. In *OPENARCH '99*, pages 90–97, 1999.

[21] M. A. L. Thathachar and P. S. Sastry. *Networks of Learning Automata*. Kluwer Academic Publishers, 2004.

[22] Cem Unsal. *Intelligent Navigation of Autonomous Vehicles in an Automated Highway System: Learning Methods and Interacting Vehicles Approach*. PhD thesis, Virginia Polyechnic Institute and State University, January 1997.

[23] Kerry Wood and Luiz A. DaSilva. Optimal max-min lifetime routing of multicasts in ad-hoc networks with directional antennas. In *2nd International Conference on Broadband Networks (BROADNETS 05)*, October 2005.