Rethinking Networking Architectures for Cognitive Control

Karen Zita Haigh, Talib S. Hussain, Craig Partridge, Gregory D. Troxel
BBN Technologies
10 Moulton Street, Cambridge, MA 02138
{khaigh, thussain, craig, gdt}@bbn.com

Abstract

The field of adaptable communication networks is a rich application area for artificial intelligence technology. Recent developments in software defined radio technology have opened up the opportunity to develop networks that are, in principle, highly adaptable and effective under a much wider range of operating conditions than currently possible. Now is the time to work with the networking community to ensure that network architectures that actively support intelligent control are designed in such a way that they will be adopted by the networking community.

We present a design for a network architecture—developed collaboratively by AI and networking researchers—that exposes significant portions of the network for cognitive control in a robust, consistent manner. We also identify some of the cultural issues that arise due to differences in the approaches of the networking and artificial intelligence communities.

Our project is not only the first networking architecture for network modules to expose internals to a cognitive controller, but also the first demonstration of cognitive control in a real-world (not simulation) mobile network. [1]

Introduction

The demand is increasing for networking technologies that support robust communication and functionality under challenging operating conditions. Network configurations are currently hand-tuned and remain static during operations. However, since user needs and operating conditions both change over time, *cognitive networks* must be designed that are be aware of their performance needs, determine if their needs are being met, and revise system configurations to better meet their needs. (Note that AI-style cognition is only one method to build a cognitive network.)

In the Adaptive Dynamic Radio Open-Source Intelligent Team (ADROIT) project, we designed a network architecture that supports both *real-time composibility* of the network stack, and *cognitive control* of the network for cognitive radio teams.

The ADROIT architecture provides rich support for cognitive applications, and allows the creation of a system that recognizes that the situation has changed and adapts the node for improved performance; anticipates changes in networking needs and plans appropriate changes in configuration for improved performance; and is able to make the appropriate changes needed in any module within a node, at any level of the networking stack and across multiple nodes of the network. This paper describes ADROIT's architectural design, highlighting the main components that allow a cognitive application to monitor and control a network stack. We also summarize key discoveries and lessons learned about how networking and AI researchers should collaborate to realize cognitive networks.

Recent developments in software defined radio technology have opened up the opportunity to develop networks that are, in principle, highly adaptable and effective under a much wider range of operating conditions than currently possible. However, while these tools provide new flexibility, none have addressed the issue of how to manage or control them [1; 5; 9], instead expecting the network designer to appropriately exploit the tools.

The unfortunate aspect of these approaches is that they rely on APIs that are carefully designed to expose each parameter separately. This approach to network configuration is not maintainable, for example as protocols are redesigned or new parameters are exposed. It is also *not amenable to cognitive control*. One issue is that there is no way to get a "directory" of the parameters that can be observed or controlled. Another issue is that there is no coordination mechanism: what happens if a cognitive controller wants to set the same parameter that another module wants to set? ADROIT is the first architecture that supports changes to existing protocols and the addition of new protocols, *without changes to other existing components*.

Most of the related work for adapting network behaviour falls under the hat of cross-layer optimization, which has a tremendous amount of literature. They generally adjust only one (occasionally two) parameters, usually with hand-built models, e.g. [6; 7; 10]. There are a few approaches that learn models of the parameter interactions, e.g. [2; 13; 17], but even these have limited numbers of parameters. Many of these approaches face two additional limitations: they are used only in simulated environments, and they do not support node mobility. Applications and protocols developed for the fixed, wired environment do not adapt transparently to the mobile, wireless environment [4]. ADROIT's architecture was designed to support the control of any number of parameters and use any kind of model to control them. ADROIT is also the first known cognitive control mecha-

[1] This paper is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under contract number NBCHC050166. Any opinions, findings and conclusions or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA); or its Contracting Agent, the U.S. Department of the Interior, National Business Center, Acquisition & Property Management Division, Southwest Branch.

nism that was demonstrated in a real-world networking system (not simulation); it used neural networks to dynamically control radio behavior for a team of mobile nodes [16].

AI Challenges in Networking

The goal of a cognitive network is to optimize the overall behavior of the distributed nodes in the network, for a given objective function, determined by current needs, e.g. as in [8]. Network effectiveness quantifies how well the network satisfies application and node requirements from mission, situational, and social standpoints. It can include a wide variety of issues including bandwidth, application-level quality of service, energy, network connectivity, and security.

There are many interesting challenges for AI in networking. The most immediate one is the massive scale involved: there are roughly 600 observable parameters and 400 controllable parameters (possibly continuous-valued) to configure *per node*[2]. We thus have a distributed, low-communication, partially-observable, high-latency optimization problem of approximately μ^{PN} choices per timestep[3]; one second would be a large timestep. Characteristics that make this an interesting domain for AI include:

Low-communication: Nodes cannot share all knowledge with all other nodes; it would overwhelm the network.

Partially-observable: Many factors that affect communication can not be observed. Few radios, for example, have a "fog" sensor.

High-latency: Many actions cause a delayed effect. For example, data transmissions from one node may only affect downstream nodes; the result takes time to propagate back to the first transmitter.

Ambiguous observations: Detection and understanding of a change in situation is not always simple. For example, how does the system automatically tell the difference between short-term fade versus entering a building?

Complex interactions: Networking parameters have deep, poorly-understood interactions with each other and with system performance. In many cases, specific pair-wise interactions can be identified, such as increased power reduces battery life. However, most of these pair-wise interactions are carefully caveatted by the networking community, with conditionals that are rarely observable or computable. Cognitive control in the general case is therefore seldom simple: the level at which symptoms appear may not be the level at which changes to the node configuration must be made; symptoms may be ambiguous at one level or at a given time and require more context; changes at one layer may impact other layers and may cause new issues; and the timing of changes may be critical.

Complex temporal feedback loops: Within a node, certain activities occur at very rapid speeds (e.g., between the Medium Access Control (MAC) and Physical layers) requiring very a very tight feedback loop to support cognitive control. Other activities (e.g., at the Routing layer) occur on a longer time-scale and cognitive control algorithms may need to take into account a wider range of factors in a slow feedback loop. Between nodes, there is yet a longer feedback loop between changes that are made and the effects that are observed in network-level performance. The variety of temporal loops and their dramatic speed differences means that correlating cause and effect of actions is particularly challenging.

In addition to the natural complexities of the domain, human users complicate the problem by requiring a certain quality of service for their application. Multiple users have interacting requirements and policies, thus creating a complex multi-objective function [8] that captures mission, situational and social standpoints.

Multi-node coordination is a particularly interesting challenge. There is a very strong norm in the networking community that all nodes must be designed and (statically) configured to interoperate. Further, typical ad hoc network or radio projects build a group of homogenous nodes. ADROIT represents a *radical* departure from this stance, in that each node has an independently operating cognitive controller, and thus network nodes *may be heterogeneous, and may be in non-interoperable configurations.* Independent cognitive controllers allow nodes to make decisions on a rapid timescale (based on locally observable values), even when the network is partitioned.[4] Meanwhile traditional AI has always assumed that that communication is "safe," negotiating and coordinating only the application-level tasks [11; 12; 18]; moreover they also generally require very high communications overhead. Ensuring that multiple nodes are coordinated enough to maintain basic communications is a key research area for cognitive networking.

ADROIT Architecture Overview

The ADROIT project focused on designing the architecture to support cognitive control of the stack. Central to our architecture is the concept of networking modules, where a traditional network layer maps to multiple modules in our system. This required major modifications to how networking protocols function, and a rich treatment of how modules interact with each other and other system components, as illustrated in Figure 1. Application data flows through the collection of network modules. Application data flows down from the applications, through the networking protocols and the radio and out into the ether. Similarly, data flows up from the the radio and networking protocols to the applications.

Network modules form the basic element of ADROIT. A module encapsulates a portion of a networking protocol that represents an important property or behavior to observe and/or an important property or behavior to adjust. The network modules work together to implement the particular type of radio required for the applications to do their work. The ability to change and reconfigure modules is central to ADROIT's flexibility, and a module may be added, removed,

[2]No current system exposes all; the highest known is about 100 parameters, of which 30 are controllable.

[3]P = number of parameters, N = number of nodes, and μ is the average number of values that a parameter can take.

[4]The alternative is to have one cognitive controller for several nodes; while coordination issues are reduced, communication overhead increases dramatically and intelligent control is vulnerable to network partitions.

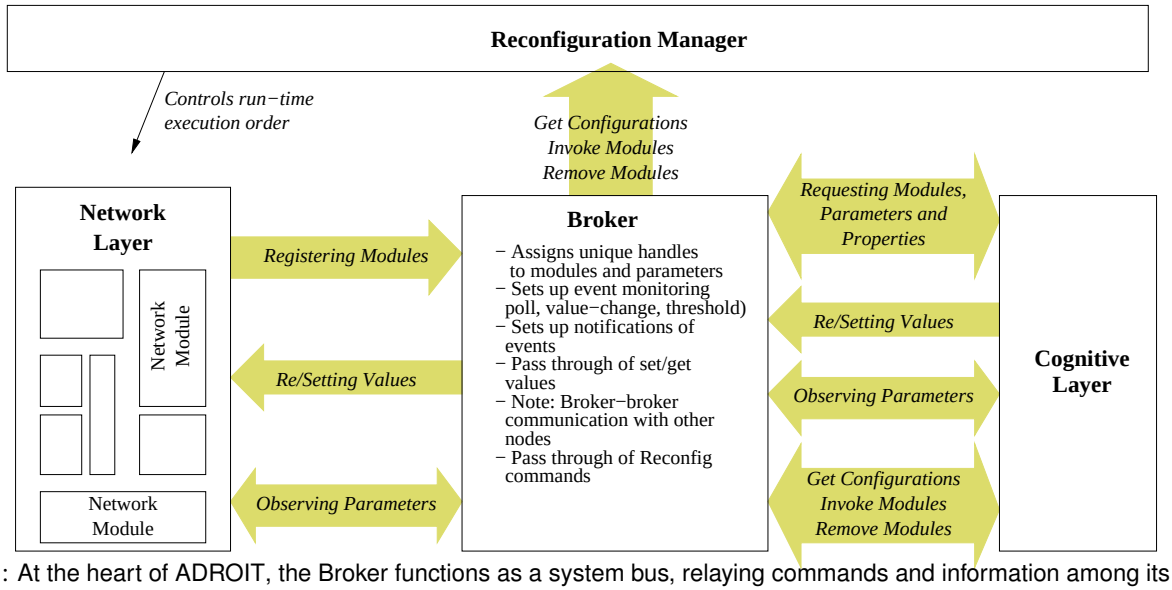

Figure 1: At the heart of ADROIT, the Broker functions as a system bus, relaying commands and information among its clients.

started or stopped while the radio is running. In a running system, each active module is an instance (or "Invocation") of a specific type (or "Implementation "). A given Implementation defines a suite of observable and comtrollable parameters. These parameters are revealed to other components in the system, such as the Cognitive Layer and other network modules, via the Broker.

The network modules are directly managed by two entities: the Reconfiguration Manager and the Broker. Network modules are indirectly managed by clients of the Broker.

The **Broker** serves as a kind of system bus between the modular software and any entity that wishes to change how a *running* module Invocation behaves. So, for instance, if a timer parameter is to be adjusted, that request is sent to the Broker, which passes the request on to the relevant module Invocation. Communication goes through the Broker even for module-to-module requests. Anything that wishes to observe, monitor, or change the state of an ADROIT radio will do so via a command relayed by the Broker. Furthermore, the Broker will notify interested parties of any changes in the radio's state or configuration.

The **Reconfiguration Manager** is responsible for creating working configurations of network modules. In general, a valid configuration may be imagined in terms of a traditional network stack, but where each layer contains one or more high-level modules that each contain sub-modules nested to an arbitrary depth. The architecture also allows for more radical arrangements of modules in principle (i.e., blurring lines between the layers). Different Implementations define different functional and data flow requirements & capabilities, and the Reconfiguration Manager will ensure that the Invocations in a proposed configuration together satisfy those requirements.

The **Cognitive Layer** is a collection of cognitive controllers that manage the radio's behavior, based on their evaluation of the radio's performance, information from applications about how applications perceive the radio's performance, and perhaps information from cognitive controllers on other radios.

A cognitive controller needs to improve the performance of a highly complex, partially observable, dynamic system in a distributed enviroment. Based on this vision, the network modules and the Cognitive Layer play different, but cooperative functional roles. Table 1 discusses some of the key roles, categorized using OODA loop terminology [3]. Note that we break out the core Observe capability into two categories: state and performance, highlighting the difference between internal module properties that are significant and relatively stable, versus moment-to-moment variations in internal module activity that capture important trends and indicate potential events of interest. The table also outlines some of the shared knowledge representations that may be required for effective coordination. These are meant to be illustrative and indicative, rather than exhaustive.

Figure 2 illustrates the ADROIT system we designed [14; 15]. The network layers correspond to traditional network layers, but comprise a number of modules which together provide the layer's functionality. All modules provide observable and/or controllable parameters that the Cognitive Layer and/or other modules have access to (via the Broker).

The Broker

The Broker mediates among the various modules of the ADROIT radio system and between modules and the applications such as network management and Cognitive Layer that wish to observe or manage the modules. Inspired by AI's agent-based systems and blackboard architectures, the Broker solution may seem familiar. However, for the networking community it represents a radical departure from traditional approaches. While the networking community has long exported named parameters for network and host management, the tradition has been to have a single writer of those parameters, for the writer to be human or a proxy for a human, and for parameter changing to be fairly rare.

Table 1: Roles within the ADROIT cognitive architecture

	Network Module Role	Cognitive Layer Role	Shared Knowledge
Observe (State)	• Inform the Cognitive Layer about current and anticipated state. (e.g., expose internal state, pollers, reachback) • Indicate whether there is outstanding activity that affect decisions (e.g., current resource commitments)	• Provide modules with guidance on important state elements (e.g., define goal, setup up listeners) • Collect patterns of state from potentially multiple sources. • Set observation frequency	• Ontology of state elements • Defined descriptors for reporting state elements (scalar, categorical)
Observe (Performance)	• Provide means for informing the Cognitive Layer about current performance profile (positive results and deficiencies). • Inform Cognitive Layer about key performance dimensions (e.g., most important parameters)	• Provide modules with guidance on performance objectives (set performance goals and target levels). • Collect patterns of activity from potentially multiple sources. • Track performance trends	• Categories of performance goals • Generic types of measures (#s & categorical) • Categories of performance dimensions
Orient	• Provide Cognitive Layer with internal analysis results (e.g., single self-reported health value, provide accessor to internal evaluator) • Provide estimates of the potential sources of problems	• Evaluate information from multiple sources • Interpret current observations • Identify potential factors influencing current observations • Anticipate future activity	• Overall health value (# vs categorical) • Cognitive-technique-specific knowledge representation (e.g., conditions on rules-of-thumb)
Decide	• Provide Cognitive Layer with possible configuration changes (e.g., dependencies, parameters)	• Determine new configuration (Decide new parameter values or change module connectivity) • Indicate reasons behind decision	• Format for a reconfiguration spec (e.g., list of parameter/values) • Ontology for describing likely effect of a re-configuration
Act	• Set new value of parameter • Attempt to achieve a goal • Implement a reconfiguration • Report the completion of a reconfiguration (e.g., success, illegal)	• Initiate reconfiguration (initiate setting of new parameter values or changing of module connectivity)	• Categories of results for a reconfiguration (success, partial, illegal, postponed)

The Broker therefore set out to solve two difficult problems in networking communication. The first is that we wish to have *one consistent interface* to any and all network modules so that if it changes, or when additional modules are created, none of their controlling applications need to be modified (including other modules in the network stack, applications, the Cognitive Layer, or even the user via a command-line interface). The second problem the Broker solves is that of *coordination of control*. Multiple controllers may be actively seeking to manage modules at the same time, and to avoid control battles, they need to know about each other.

To perform these services, the Broker implements an interface that supports the following functions:

Directory Services: Implementations and Invocations register themselves with the Broker. Registration is the process of notifying the Broker that they are present and available. Modules are required to describe themselves, their configuration dependencies, their expose parameters. It is possible to ask the Broker to search for active Invocations based on their Implementation or type.

Parameter Management: The Broker views every invocation as containing a suite of parameters. These parameters may be read, monitored (e.g. to determine if they change to a value outside expected limits), and some may be altered (to change system behavior).

General Message-board architecture: The Broker provides a message-board in which messages are posted to and read asynchronously. The purpose is to allow any Invocation to track what other Invocations are doing.

Configuration Management Pass Through: The Broker is not responsible for configuration management. However, because so many of the Broker's clients need to know the configuration and also for simplication of APIs, the Broker maintains a pass-through interface, in which requests regarding configuration (including checking the viability of proposed new configurations) are passed through to the Reconfiguration Manager.

All clients of the Broker use the same interface to expose their parameters and to read or set parameters of other clients. A module may *not* expose its parameters for outside control via any interface except the Broker. This restriction allows the Broker to consolidate requests and control thrashing (e.g., if multiple clients wish to control the same module in different incompatible ways). The Broker thus provides as much visibility or control over the modules as they offer via their exposed parameters.

Calls to and from the Broker are asynchronous so that no real-time network module is blocked while waiting for a response from the Broker. A request to change a parameter or configuration is acknowledged by the Broker on receipt, but the actual change may take place some time later. This requirement was key to making ADROIT's new architecture acceptable to the networking community. From the cognitive control side, there were also advantages. Changing the state of the radio might require changing dozens of parameters and the fastest way to achieve this change is to issue

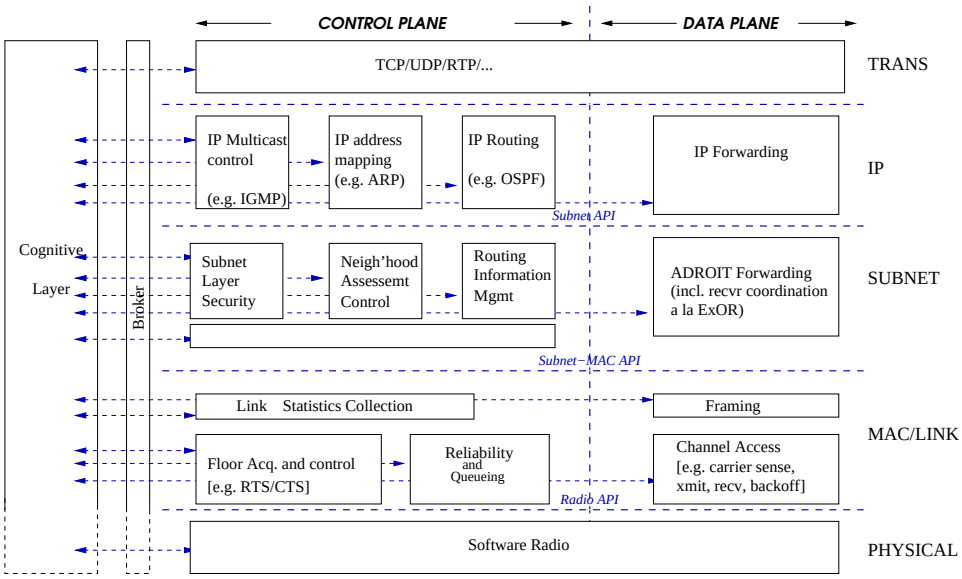

Figure 2: Cognitive control across modular networking stack.

change requests for all the parameters concurrently and then confirm the changes as they actually take place.

Cognitive Control within ADROIT

The ADROIT architecture provides a rich suite of opportunities for cognitive control across the networking stack, and places few limits on the types of techniques used to perform the control and the motivations for control. For instance, *rules of thumb* or relationships among parameters may be easily captured and exploited within ADROIT. This knowledge could be used for a variety of purposes, including state-space reduction, feature construction, and guidance for configuration, optimization, control and experimentation. Via real-time access to observable parameters, a *statistical learning* technique can gather high-fidelity data about system performance and learn to categorize/recognize key changes in behavior or operating conditions. *Distributed planning* techniques may be used to coordinate the configurations of different nodes to maximize connectivity and throughput. For instance, the communication frequencies and/or protocols used may be adjusted collaboratively based on performance feedback.

We have developed a number of walkthroughs based on the system design of Figure 2 to flesh out potential cases for cognitive control. We summarize two of these cases here to highlight the possible uses of different AI techniques within ADROIT. More details and additional walkthroughs exist elsewhere [15].

Walkthrough: Minimize Retries

Consider a Reliability Control and Queueing Module in the MAC layer that uses an Automatic Repeat-reQuest (ARQ) algorithm for error control with a single ACK for a data burst. It could expose the following parameters:

- **Observable**
  - Percentage of frames dropped
  - Number of requested re-transmissions
  - Average confirmation time (i.e., round-trip for ACK)
- **Controllable**
  - Maximum confirmation time to wait before re-trying
  - Maximum number of retries before dropping

Also consider a Link Characterization Module in the Subnet layer that collects observations about the performance of the links the node has to its neighbours, including

- **Observable**
  - An estimate of the number of transmissions needed to reach the neighbor
  - Percentage of frames received by a neighbor within a large time window

Over time, due to changes in factors, such as frequency of transmission, environmental effects, distances among nodes, congestion, etc., the round-trip time for a transmitted frame/ACK may increase resulting in confirmation times that are consistently above MaxConfirmationTime. If this happens, then the node will be consistently re-sending frames more than once on average, unnecessarily.

Based on observations from these two modules and knowledge about the state of local queues, a cognitive controller in the Cognitive Layer can determine how many re-tries are required on average for each neighbor and across all neighbors, and can infer whether earlier frames are generally received or lost. If the average number of re-tries increases significantly without corresponding frame loss, an opportunity for corrective action occurs: a sufficient increase in MaxConfirmationTime should result in a decrease in the number of re-tries. If queues fill and frames are dropped consistently, then there is another opportunity: a sufficient decrease in MaxConfirmationTime should reduce drops.

To continually minimize the number of retries, the cognitive controller could use various AI techniques, such as

rules-of-thumb (e.g., increase MaxConfirmationTime incrementally by 10% until there is a decrease in number of retries), or statistical learning (e.g., learn patterns of behavior based on past performance and associate key patterns with successful corrective actions).

Walkthrough: Maintain Accuracy of Knowledge

Consider a Neighborhood Assessment module that applies an algorithm based on collecting heartbeats over a window of time in order to assess whether a link is up or down. Such a module could expose the following parameters:

- **Observable**
  - Total number of heartbeat points received in window
- **Controllable**
  - HeartbeatInterval: Number of seconds to wait between sending heartbeat messages
  - WindowSize: Number of seconds over which to collect heartbeats to determine if a link is up or down

As events occur within the world and/or within the network, connectivity between nodes can change. If a high frequency of heartbeats is used, accurate knowledge about links can be maintained but a high overhead traffic is incurred.

Based on observations from this module as well as knowledge about the state of local queues, a cognitive controller in the Cognitive Layer can determine the frequency with which link status changes occur. If the frequency has increased significantly and remained high for a prolonged period of time, and if local queues are getting bigger, there is an opportunity for corrective action: a lower HeartbeatInterval can result in more responsive computation of link status, better maintenance of bi-directional links, more effective routing decisions and smaller queues (since we are less likely to be waiting for a down link). If the frequency has decreased significantly and remained low for a prolonged period of time, then there is another opportunity: a higher HeartbeatInterval can reduce network overhead with minimal effect on link status maintenance.

The cognitive controller would help maximize information sharing and multi-node coordination by minimizing routing via down links. It could also coordinate with other cognitive controllers (e.g., for communication frequency used) to maximize network connectivity.

Networking and AI: Cultural Issues

There were several revelations and lessons learned during the development of the architecture related to the differences in the approaches used by networking and AI practitioners. We identified numerous technical and cultural issues.

Benefits and scope of cross-layer design. Most networking people are familiar with "cross-layer design." However, within the networking community, this concept usually means two layers, and one or two parameters in each layer. Networking people were generally somewhat skeptical about how much benefit multi-layer coordination would bring. Together, we developed detailed drill-down walkthroughs, each focusing on how certain changes in parameters could produce novel changes in networking protocols and behavior under certain observed conditions. The walkthrough process benefitted both groups, with networking people becoming more accepting of system-wide cognitive control, AI people understanding more of the reasons for traditional networking approaches, and everyone having a better understanding of how to make the new approach work better.

Relinquishing control outside the stack. Networking people were very concerned about an outside controller making decisions about performance. AI researchers meanwhile, like to say "give me everything and I'll give you the answer." Neither extreme is appropriate. We therefore added several technical modules that helped us meet in the middle, including "failsafe" mechanisms that would allow a network module to reject things that didn't make sense, and constraint publishing & checking mechanisms that would require the Cognitive Layer to keep settings consistent. The walkthroughs also helped both sides to see the concerns and benefits.

Reliance on a centralized Broker. Parts of the broker are similar to traditional network management, but the new system architecture called for the broker to be a mandatory part of the system, with failure of the broker having grave consequences. Networking design has tried to minimize the number of components that must be relied upon, and relying on something "cognitive" (and therefore complicated and not entirely predictable) was viewed with particular suspicion. Therefore each network module was expected to have a failsafe default operation that would work, even when the Broker was not functional.

Asynchrony and Threading The AI people were used to programming in languages such as Java where threaded operation was the norm, and programs waiting for an answer might make a blocking call in a worker thread. The networking people understood this model, but outright rejected an approach that would require network modules to be written in this manner. Much networking code is written in a single thread that must make only non-blocking code, and it is often in an OS kernel with a reduced programming environment. We implemented the Broker so that the client library to be linked with networking modules could be used by non-threaded programs.

Boundaries. In traditional networking approaches, there is a very clear boundary between application and network module — often corresponding to a user/kernel boundary with a widely known API (e.g., "BSD sockets"). Similarly, there is a clear boundary between controller and controllee. With the generic approach to exposing and controlling parameters, these boundaries blur. Any client of the Broker can choose to expose controllable parameters, and any client can choose to set another module's parameters. Thus, an application can choose to have complete visibility into the stack, or be told to back off by the network. While both groups believed that better performance could be achieved, the AI people focused more on the benefits of flexibility and the networking people more on their concerns that complexity would lead to unreliable systems. We expect that removing traditional restrictions will result in interesting and significant new ideas.

Heterogeneous and *non-interoperable* nodes. A deeply-held tenet in networking is conformance to written protocol specifications; all nodes must always follow the protocol, and from this one can conclude that they will interoperate (but one cannot guarantee maximal performance). Further, most nodes are homogeneous. Cognitive controllers enable the network to become heterogeneous to the point of non-interoperability, resulting in possible failure of the nodes to communicate, but also enabling greater performance when managed correctly. To address the mandate of maintaining connectivity, while allowing for heterogeneity, our architecture includes an"orderwire" bootstrap channel to be used when a node can not communicate with the rest of the network. We planned to explore several different mechanisms for coordinating the heterogeneity.

Conclusions

ADROIT is a unique system that combines software radios with cognition. A very detailed architecture description appears in [14; 15]. Vital to making ADROIT work is a modular protocol architecture with a generic API that exposes parameters and modules for cognitive control in a consistent, maintainable manner. The two most significant impacts of our work are:

- The implementation of the first networking stack architecture that supports changes to existing network protocols, such as exposing a new parameter, and the addition of new protocols, *without changes to other existing components.* This new design ideally supports the addition of cognitive control modules.
- The implementation of the first known cognitive control mechanism that was demonstrated in a real-world networking system (not simulation). It used neural networks to dynamically control radio behavior. Results are presented in [16].

The domain of cognitive networks is challenging and interesting for AI researchers; it has many poorly understood complex interactions, but is well-sensored and ripe for expanding AI concepts.

Acknowledgements: The authors would like to thank Jonathan Smith and Lee Badger of DARPA for their support and insights. We would also like to acknowledge the contributions of our fellow BBNers, authors of [16], to the ADROIT system design and their commitment to the challenge of incorporating cognitive control into the network.

References

[1] E. Blossom. GNU radio: tools for exploring the radio frequency spectrum. *Linux Journal*, 2004.

[2] J. A. Boyan and M. L. Littman. Packet routing in dynamically changing networks: A reinforcement learning approach. In *Advances in Neural Information Processing Systems (NIPS)*, pages 671–678. Morgan Kaufmann, 1994.

[3] J. R. Boyd. An organic design for command and control. In *A Discourse on Winning and Losing*, 1976. Unpublished lecture notes.

[4] W. W. Brown and T. Krout. Future performance expectations for mobile wireless communication networks. In *AFCEA meeting*, San Diego, CA, January 2006.

[5] A. Casimiro, J. Kaiser, and P. Verssimo. An architectural framework and a middleware for cooperating smart components. In *Proceedings of the 1st Conference on Computing Crontiers*, pages 28–39, Italy, 2004.

[6] Q. Chen and Z. Niu. A delayed adaptive retransmission scheme for false route failure in MANET. In *Proc. 5th International Symposium on Multi-Dimensional Mobile Communications*, pages 858–862, 2004.

[7] H. Gharavi and K. Ban. Cross-layer feedback control for video communications via mobile ad-hoc networks. In *Proc. IEEE 58th Vehicular Technology Conference*, pages 2941–2945, 2003.

[8] K. Z. Haigh, O. Olofinboba, and C. Y. Tang. Designing an implementable user-oriented objective function for MANETs. In *IEEE International Conference On Networking, Sensing and Control*, U.K., April 2007.

[9] M. A. Hiltunen and R. D. Schlichting. The cactus approach to building configurable middleware services. In *Proceedings of the Workshop on Dependable System Middleware and Group Communication (DSMGC)*, Germany, October 2000.

[10] P. Lettieri and M. B. Srivastava. Adaptive frame length control for improving wireless link throughput, range and energy efficiency. In *Proc. International Conference on Computer Communications INFOCOM (2)*, pages 564–571, 1998.

[11] T. W. Malone and K. Crowston. The interdisciplinary study of coordination. *ACM Computing Surveys*, 26(1):87–119, March 1994.

[12] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2):149–180, 2005.

[13] C. Monteleoni, H. Balakrishnan, N. Feamster, and T. Jaakkola. Managing the 802.11 energy/performance tradeoff with machine learning. 2004.

[14] G. D. Troxel, E. Blossom, S. Boswell, A. Caro, I. Castineyra, A. Colvin, T. Dreier, J. B. Evans, N. Goffee, K. Z. Haigh, T. Hussain, V. Kawadia, D. Lapsley, C. Livadas, A. Medina, J. Mikkelson, G. J. Minden, R. Morris, C. Partridge, V. Raghunathan, R. Ramanathan, P. G. Rubel, C. Santivanez, T. Schmid, D. Sumorok, M. Srivastava, R. S. Vincent, D. Wiggins, A. M. Wyglinski, and S. Zahedi. Enabling open-source cognitively-controlled collaboration among software-defined radio nodes. *Computer Networks*, 52(4):898–911, March 2008.

[15] G. D. Troxel, S. Boswell, A. Caro, I. Castineyra, A. Colvin, Y. Gabay, N. Goffee, K. Z. Haigh, T. Hussain, V. Kawadia, D. Lapsley, C. Livadas, A. Medina, J. Mikkelson, C. Partridge, V. Raghunathan, R. Ramanathan, P. Rubel, C. Santivanes, D. Sumorok, B. Vincent, and D. Wiggins. Adaptive dynamic radio open-source intelligent team (ADROIT): Architecture and design. Technical Report BBN-TR-TBD, BBN Technologies, 2007.

[16] G. D. Troxel, A. Caro, I. Castineyra, N. Goffee, K. Z. Haigh, T. Hussain, V. Kawadia, P. G. Rubel, and D. Wiggins. Cognitive adaptation for teams in ADROIT. In *IEEE Global Communications Conference*, Washington, DC, November 2007. Invited.

[17] T. Ye, H. T. Kaur, and S. Kalyanaraman. Large-scale network parameter configuration using an on-line simulation framework. In *IEEE/ACM Transactions of Networking*, 2003.

[18] X. Zhang, V. Lesser, and S. Abdallah. Efficient Management of Multi-Linked Negotiation Based on a Formalized Model. *Autonomous Agents and Multi-Agent Systems*, 10(2):165–205, 2005.