Cognitive Adaptation for Teams in ADROIT

Gregory D. Troxel, Armando Caro, Isidro Castineyra, Nick Goffee, Karen Zita Haigh,
Talib Hussain, Vikas Kawadia, Paul G. Rubel, David Wiggins

BBN Technologies, Cambridge, MA 02138 USA Email: gdt@bbn.com

***Abstract*— We have created a sensor-sharing protocol that uses cognition to increase performance by choosing protocol parameters based on the current environment and the past relationships between environment and performance. We have constructed a prototype of the protocol, and experimented with it in a four-node outdoor testbed. Our testbed is part of a larger effort, ADROIT, which seeks to create cognitive teams of software-defined radios [1].**[1]

I. Introduction

The Adaptive Dynamic Radio Open-source Intelligent Team (ADROIT) project [1] is a large effort building teams of cognitive, open-source, software-defined radios (SDRs). ADROIT has built the initial parts of a software-defined data radio that is intended, from the start, to be controlled cognitively.

The ADROIT definition of a cognitive radio differs from the classic definition of Mitola and Maguire [2]. In their definition, cognition is internal to the radio. In ADROIT, both programs using the radio and cognitive controllers are cognitive, but the radio itself is not necessarily cognitive. This difference implies that the radio must expose its internal workings so that external entities can manage the radio's behavior.

As part of ADROIT, we developed a sensor-sharing protocol capable of being controlled cognitively, and a learning-based method using neural nets to optimize the performance of the team. After briefly summarizing the past literature, we explain our cognitive approach, our implementation, and our experimentation results.

II. Prior Work

There have been suggestions to embed cognition into network management [3], as well as efforts to allow cognitive entities to mediate between applications and a balky network [4].

There has also been a recognition that cognition is extremely well-suited to network management of SDRs. DARPA's Next Generation (XG) project found that cognitive tools were essential to allowing the radio to decide which frequencies were free and how to best exploit them. (Part of the issue for XG was that the availability of a frequency was determined not just by what traffic could be found at that frequency, but also by a complex set of FCC rules dictating how the frequency could be used.) In a prior experiment at BBN, we found that a cognitive tool using genetic algorithms was far better at configuring a software radio with over a thousand configuration options than the best-trained radio engineers.

Distributed AI techniques (e.g., [5], [6]) require massive communications with non-neighbors, and universally do not support mobility (changing connections or constraints between the nodes). Control-theoretic approaches are inappropriate for nonlinear systems such as MANET [7].

Neural nets have been used to configure parameters of a MANET to improve throughput in simulation [8]. Our work follows the approach of Haigh, Varadarajan, and Tang, but differs by optimizing application-layer performance. Further, it was done in a testbed rather than via simulation.

III. Problem formulation

We describe the problem that our system was asked to solve, and briefly describe our testbed. The problem is representative of actual problems, and the formulation seeks to place the system in a stressed regime in order to evaluate learning.

A. Communication

The system consists of four mobile nodes and one stationary node. The mobile nodes communicate locally; our system used IEEE 802.11 Independent Basic Service Set (IBSS) mode. Two mobile nodes also have a wide-area connection to send data to the stationary node; our system used EVDO service from two carriers. While our stationary node was at the same location, the network path went from Cambridge, Massachusetts, to New York City and back. The motivation was to emulate a low-capacity connection from the team to a remote node.

B. Mobility Patterns

The four mobile nodes roam in a 100 m by 100 m area, which is logically divided into 1 m by 1 m grid squares.[2] The nodes follow preplanned paths which are mostly disjoint, so that about 20% of the grid squares are visited by each node, and about 80% are visited in total, with minimal overlap.[3] The paths avoid grid squares with obstructions that make it impossible or difficult to visit. While the paths are preplanned, the measurement and communication software is unaware of the paths. Each 30-minute traversal of paths is called a run.

[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.

[2]Our test area was 135 m by 80 m due to site limitations.

[3]Our paths were between 1910 and 2450 m, resulting in average speeds near 1.2 m/s. The sum of lengths was 8713 m, with 8327 total squares visited.

Fig. 1. ADROIT wagons before a test run.

C. Sensor Measurements

There are two 802.11 access points (APs) at different locations near the grid, the sole purpose of which is to provide beacons for the mobile nodes to measure. The task is to measure the signal strength of each AP at each grid square, and to share this information with all four mobile nodes and the stationary node. Note that the signal strengths of the APs are not directly related to signal strengths on the inter-node channel. The experiment takes 30 minutes for data collection, followed by 10 additional seconds for sharing the results among the four mobile nodes and the stationary node.

D. Hardware Configuration

Each of the mobile nodes comprises a computer running NetBSD, two 802.11 interfaces (for internode communications and for measuring the signal strength of the APs), a GPS receiver, associated batteries and antennas, and a wagon for transporting the equipment (Figure 1). The stationary node is a computer with a normal wired Internet connection. Our system also includes an additional node whose only role was to generate local GPS differential corrections, thereby providing the mobile nodes with submeter position accuracy.

E. The Challenge

The basic goal is for each of the four mobile nodes and the stationary node to have an observed signal strength value for as many grid squares as possible. The metric is defined as $\mathrm{MG} = \min_i\{s_i\}$ where s_i is the number of grid squares for which node i has a signal strength measurement.

The second goal is to use cognitition to cause a second *operation* run (where signal strength observations from the first run have been deleted) to have observations for more grid squares than the *training* run. The metric is $\delta\mathrm{MG} = \mathrm{MG}_O/\mathrm{MG}_T$ where MG_T and MG_O are the metric, g, for the first (training) and second (operation) runs, respectively.

The experiments are not completely repeatable due to GPS positioning errors, interference from sources outside the experiment, and human error positioning the wagons. Further, the node/path relationship may be permuted, the start position of each path different, and some paths traversed in reverse. Thus, a node may visit locations that it did not visit—or even get close to—in the first run. The path permutations are done in order to force nodes to operate in situations not encountered in training. Thus, nodes must learn from the experience of other nodes in order for the team to succeed.

Given enough internode capacity, one can easily achieve 100% sharing; our system did. In order to place the system in a regime where learning could occur, we padded each signal strength report with random bytes, and always carried the padding with the report. We found that 2000 bytes of padding per measurement was necessary to cause only about 5500 out of 8000 observations to be shared during training runs. Each transmitted measurement was thus two packets due to IP fragmentation, because we used the common Ethernet 1500-byte MTU even though the native 802.11 MTU is higher. We used the same padding during operation runs.

The problem can be solved by state distribution of all observations, e.g. node n at time t was at (x, y) and heard AP a with signal strength s, or state distribution of the resulting map (x, y, a, s) without distributing observations to nodes that already have an observation for the grid square. We distributed the observations themselves.

This problem formulation differs from human learning in a key way. Humans learn at all times and when encountering a task can draw on much prior learning, even if almost all of it is not directly locally applicable.

IV. Protocol

Nodes make observations of the APs' signal strength and record them in a local database. We describe our protocol for distributing observations to all other nodes. The protocol does not assume full IP connectivity; some nodes can never communicate directly and only via a third node. In this phase, our testbed did not attempt to provide multicast routing, and thus multicast packets from a node arrive at some subset of the peers, depending on path loss, interference, and congestion. The EVDO links to the stationary node are via commercial service and thus unicast only.

The protocol essentially consists of synchronizing observations with other nodes in range. This is done by periodically (5 s) multicasting `infobeacon` messages, which compactly summarize observations present in the node's local database. Nodes receiving an `infobeacon` multicast `infobeacon replies` with observations that they have (which may have originated on other nodes) and the sender does not, up to some number of observations (typically 500). A node's own information will be sent to neighbors when the neighbors assert that they do not have it, and information from all nodes will reach all others. On EVDO links, packets are always sent unicast, because multicast is not available.

The `infobeacon` packets also contain the node's current position, and signal strengths of the internode links (*not* the APs' signal strength). Position is used for one of the protocol

variations, described below, and the internode signal strengths are used for the environment model for learning.

As an optimization for good conditions, each node periodically multicasts a `freshinfo` packet with new observations. Note that this would not be sufficient to solve the problem even if there were no loss, because packets from the nodes without the EVDO connections would not arrive at the stationary node.

Our protocol supports multiple kinds of records. In addition to observations, we create command records to start/stop the experiment, status records for observing the experiment in progress, and performance counters to aid in learning. The protocol was implemented in python.

To enable learning, our protocol has two binary strategy variables, with the cross product leading to four strategies. The strategies affect only the internode network; EVDO performance is too variable to use as the basis for experiments. We chose these particular variables because we believed that they would affect performance, but could not predict under what circumstances which strategies would be better.

The basic protocol uses only multicast packets. The first strategy choice is whether to use multicast for `infobeacon replies` or to send replies unicast to the `infobeacon` sender. We call this first strategy choice `multicast`. In 802.11 IBSS mode, multicast packets are sent at the basic rate (1 Mb/s) and are not acknowledged. Unicast packets are sent at a rate chosen by a per-recipient rate-adaptation algorithm (SampleRate [9] as found in NetBSD's `ath(4)`, which chooses rates to minimize channel occupancy time of the packet, acks, and any necessary retransmissions). Multicast gets data to multiple receivers, but is slow and can be unreliable. Unicast can be faster and reliable, but data must be sent to each recipient separately.

The second strategy choice is `farthest`. If off, the system behaves as described earlier. If on, `infobeacon replies` are suppressed if the information to be sent was not measured by the replying node and that node is farther from the requester than the other nodes (as measured using GPS and communicated by position reports in `infobeacons`). This suppression typically results in fewer packets, and thus less channel contention, but it also could hurt information sharing.

V. Learning

The problem formulation specifies experiments comprising two runs. The first *training* run has no state information. The second *operation* run, may use state information (except signal strength observations) from the first run.

Our goal is to have the nodes cooperate to improve the team's performance, rather than to have each node optimize local performance. This is dictated by the metrics, which are determined by the performance of the node with the fewest measurements (the "worst node"). We require a distributed algorithm that enables each node to choose strategies independently, and reject schemes using central controllers.

A key aspect is choosing behaviors that work well in different situations, rather than choosing a single parameter value for an entire future run. Our approach is to build, during training, a model relating environment (E) and strategy choices (S) to outcomes (O), and then to use that model to make strategy choices during operation: $O = f(E, S)$. This decomposition requires that we address environment representation, outcome definitions and measurement, and model representation.

A. Basic Operation

The system operates on a short interval (5 s), gathering metrics and choosing strategies.

During training runs, the strategy is changed once per minute according to a pre-generated list of strategies with pseudorandom order and even distribution of the four. The choice of one minute is the result of a difficult tradeoff. Faster changes allow observing how well more strategies work in a larger variety of situations. Slower changes allow more reliable attribution of behavior to strategies; it may be that after changing strategy it takes 10s for the behavior to stabilize.

During operation runs, the model is queried each interval with the current environment and each strategy. The strategy that is predicted to lead to the best outcome is used for the next interval. While in theory the same stabilization concerns are present, we found that strategy choices remained stable over long enough periods that no practical problems arose.

B. Metrics

Our problem is stated in terms of a global metric MG: the number of mapped grid squares at the node having fewest mapped grid squares at the end of the experiment, where a mapped grid square is defined as one for which at least one signal strength measurement is available at the node via local measurement or sharing by other nodes. Clearly we can observe this metric only at the end. In order to reach that goal, we define other (global) metrics and then (local) performance counters, which when shared can be used to compute the global metrics. We believe (and confirmed) that optimizing the new metrics will lead to better values of the original metrics.

We define M1 as the fraction of observations present at the worst node relative to the total observations generated. This metric measures sharing performance, reaching 1.0 for perfect operation. Note that multiple observations could be made in each grid square (by possibly different nodes). We define MG similarly, but projecting to unique grid squares at each node before choosing the worst. M2 is the number of packets transmitted divided by the number of observations present at the worst node. This metric measures sharing efficiency. Note that M1, MG, and M2 are metrics for an entire run.

We define metrics $\mathrm{m1}(t)$ and $\mathrm{m2}(t)$ that can be computed over an interval. Let $s_i(t)$ be the number of observations present at node i at time t, and $o_i(t)$ be the cumulative number generated at node i. Then $\mathrm{M1(t)}$ is $\mathrm{min_i\{s_i(t)\}/\sum_i o_i(t)}$, or the minimum shared over the sum generated. As t approaches the end, this becomes M1 above. Then, let

$$\mathrm{m1}(t) = \frac{\min_i\{s_i(t)\} - \min_i\{s_i(t-1)\}}{\sum_i o_i(t) - \sum_i o_i(t-1)}$$

This metric represents the progress made at the worst node relative to the total new observations generated. Note that

"progress at the worst node" is subtle; this is not the improvement at the node with the fewest observations at the beginning of the interval, but the difference between the fewest before and after—the worst node could change. We define $\mathrm{m2}(t)$ similarly, dividing packets sent by all the nodes in the interval by the numerator of $\mathrm{m1}(t)$.

To compute these metrics, one needs the counts of measurements generated, measurements present, and packets sent at each interval boundary. To provide access to all counters at all nodes, each node generates a performance record at the end of every 5 s interval. These records are then shared reliably via the `infobeacon` mechanism.

When a training run ends, metrics are computed for every interval for which both the beginning and ending performance records are available for all four nodes. We set the sharing priority of records as command, performance, status, and sensor data in order to deliver most performance records.

Because there are two metrics, we must impose a total ordering in order to choose a strategy to maximize the outcome. Therefore we defined a utility function on $(m1, m2)$. Our first attempt was $0.95m1 + 0.05(10 - m2)$, intending to weight m1 heavily. Operation runs resulted in better m1 but much worse m2. We defined an alternate utility function which is as above if $\mathrm{M1}(t)$ to date is < 0.95 and otherwise seeks solely to minimize m2. The intent behind this choice is that trying to maximize M1 when the system is close to achieving its goals perfectly results in responding to noise. Even with this more complex function, our system did not improve on M2.

C. Environment

We believed that the characteristics of the internode data links would affect which strategies were successful. We would like to gather and use ETX (estimated transmission count [10]) because it directly relates to packet delivery, but this was impractical. We used the signal strength of received packets.

We desire a global environment representation, and start with the matrix of node-node signal strengths. The diagonal is missing, leaving twelve values. This representation is problematic because it does not recognize that two situations that differ only in node numbering are the same. We therefore normalize the environment by sorting the other nodes by received signal strength, carrying higher-order terms in the list of twelve along as we sort. Thus we have $r_h, r_m, r_l, t_h, t_m, t_l$, where r_x is the received signal strength from a node and t_i the strength of our signal received at that node. Following we have the links between other nodes, i.e., $r_{hm}, r_{mh}, r_{hl}, r_{lh}, r_{ml}$ and r_{lm}.

Reliably transferring environment measures would introduce latency. This latency would not harm the training run, but would result in using old data during operation runs. Further, the data used in operation runs would have a different processing path from the data in training runs, and this seems likely to introduce distortions. Therefore, whenever we need an environment representation, we use the best one available, and use the same processing path for training and operation.

D. Neural Net for Model Representation

We use a neural net to represent our model of $O = f(E, S)$. The environment representation and strategy choice are twelve and two inputs, and two outputs represent (incremental) m_1 and m_2. We produce a training row from every interval in the training run, consisting of the environment at the *beginning* of the interval, the strategy used, and the outcomes. Intervals for which performance data were not available are omitted. The neural net is then trained on these data; we typically have about 360 training rows. Then, it is straightforward to query the neural net for a given E and all values of S, and choose the S resulting in maximal O.

It is far less straightforward to choose neural net structure and parameters to ensure that useful answers are obtained without overtraining. We separate the data into 80% training and 20% validation by randomly selecting rows from the entire data set, with the constraint that the distribution of strategy choices is even. Multiple nets are trained with the training data, and then their predictions on the validation data are compared with the observed outcomes. The system chooses a net with the best accuracy on the validation data. We used the Fast Artificial Neural Network (FANN) library.

Our neural net selection and training is automatic. At the end of a training run, the nodes train multiple nets and validate them, and save the net representation in a file to be used by the operation run. We do send a start command to start the operation run, but it contains essentially only the start time, padding level, and that the run should be an operation run.

VI. Results

Our primary metric was the number of grid squares present at the node having the fewest. We conducted several experiments on different days, and report the valid outcomes. Unfortunately, resource limitations prevented us from performing more tests. Some experiments were considered invalid due to interference to our GPS receivers; these are characterized by either excessive observations (due to perceived motion to grid squares that were not in fact visited) or very few observations (due to not having valid position data). Each test consisted of a training and an operation run. The primary metric improved by about 10% due to learning (Table I). In some experiments, M1 increased, but in some it did not. The interaction between MG and M1 is complex. Some times the operation run generates more observations than the training run, and the combination of available channel occupancy time and padding constrains the amount of data sent rather than the fraction of available data. Still, maximizing M1 during a particular run leads to delivery of more data and thus higher MG.

TABLE I
Results from two tests

Test	Training	Operation	Improvement
MG = Grid squares present at the node having the fewest			
Test1	5532	6116	10.55%
Test2	5685	6146	8.10%

Our improvement of roughly 10% is more significant than it appears. Each of the nodes makes roughly 25% of the observations, and 10% of the minimum observations present is more like 15% of the observations sent from one node to other nodes. The system is stressed, but because the internode ranges vary greatly, the system's throughput can be expected to vary greatly. Progress made during good times will dominate average progress, and a metric that tries to capture the amount of incremental sharing achieved during the worst 30 seconds might well show much greater (or less) improvement.

Another interesting aspect was observing the strategies that were chosen. We found that `farthest` was chosen to be on most of the time. In hindsight, this makes sense, because our system was operated in a congested regime where essentially every node always had data to send. Thus even if this strategy resulted in delayed delivery of information, it would succeed in improving the main metric if overall throughput increased due to fewer packets with duplicate data.

The `multicast` strategy choice was far more interesting. When the nodes were all very close together (high signal strengths), unicast was preferred. We believe that this is because high data rates (up to 54 Mb/s) can be used, achieving better performance than multicast despite the factor-of-3 penalty. At medium ranges, multicast was preferred. We believe that here the gain from muliple recipients per packet dominated. At long ranges, near the limits of connectivity, unicast was again preferred. We believe that this is because multicast data delivery in 802.11 IBSS mode is unreliable and that MAC-layer retransmissions of unicast packets is helpful.

We found that when the system was operated under unstressed conditions it did not improve on the value of the secondary metric M2 (packets per shared observation), even though M1 (fraction of observations shared) was essentially unity for both runs. We believe that the system was chasing M1 during many intervals, resulting in increased packets that dominated the final M2 value. Because catching up in M1 was easy, this effort—and associated cost—was unnecessary. It appears difficult to optimize two such long-period metrics simultaneously by choices made over short timescales with short-term performance information. A more complex model of expected performance is likely needed.

VII. Scaling

It would be easy to pick 10 strategy choices, rather than two, but our system would then not work, requiring 1024 sets of training intervals rather than four. In continuous use a system would likely run in training and operation mode simultaneously, after possibly an initial training-only period. This would result in more frequent use of previously successful strategies. Following SampleRate [9], one could occasionally try a randomly-chosen strategy. Strategies which consistently perform badly could be eliminated from consideration.

VIII. Conclusion

We constructed a system that used learning to choose protocol parameters based on the past relationship of the environment and strategy choices to outcomes. Our system produced significantly improved outcomes.

The choice of short-term metrics the maximization of which leads to long-term goals is very difficult. The choice of environment representation is very difficult; the representation must capture enough about the situation so that knowledge is applicable, but abstract out unnecessary information so that it may be applied to similar situations.

In hindsight, we can explain why the system chose strategies under various circumstances, but were not able to predict the choices. We believe that the system would adapt to different behaviors, given enough strategy choices.

There remain two very significant challenges. One is how to automate more of the learning process. If the system could generate strategy choices, explore them over a long time, and prune those that didn't work, it would likely be more effective (and more like human learning). The second challenge is scaling. Absent simulation, it takes a large amount of time to try many alternatives.

Our work used an actual testbed, rather than a simulator. Our system was able to improve performance in a complex RF environment that cannot be easily modeled.

Acknowledgment

The authors would like to thank Jonathan Smith and Lee Badger of DARPA for their support and insights.

References

[1] G. Troxel, E. Blossom, S. Boswell, A. Caro, I. C. A. Colvin, T. Dreier, J. Evans, N. Goffee, K. Haigh, T. Hussain, V. Kawadia, D. Lapsley, C. Livadas, A. Medina, J. Mikkelson, G. Minden, R. Morris, C. Partridge, V. Raghunathan, S. Ramanathan, C. Santivanez, T. Schmid, D. Sumorok, M. Srivastava, R. Vincent, D. Wiggins, A. Wyglinski, and S. Zahedi, "Adaptive dynamic radio open-source intelligent team (ADROIT): Cognitively-controlled collaboration among SDR nodes," in *Proc. First IEEE Workshop on Networking Technologies for Software Defined Radio (SDR) Networks*, 2006.

[2] J. Mitola and G. Maguire, "Cognitive radio: Making software radios more personal," *IEEE Personal Commun. Mag.*, vol. 6, pp. 13–18, Aug 1999.

[3] D. Clark, C. Partridge, J. Ramming, and J. Wroclawski, "A knowledge plane for the internet," in *Proc. ACM SIGCOMM Conference*, Aug 2003.

[4] "SAPIENT: Situation aware protocols in edge network technologies," DARPA. [Online]. Available: http://www.schafertmd.com/sapient/

[5] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo, "ADOPT: Asynchronous distributed constraint optimization with quality guarantees," *Artificial Intelligence*, 2005, to appear.

[6] W. Zhang and Z. Xing, "Distributed breakout vs. distributed stochastic: A comparative evaluation on scan scheduling," in *AAMAS-02 Third International Workshop on Distributed Constraint Reasoning*, Bologna, Italy, July 2002, pp. 192–201.

[7] H. K. Khalil, *Nonlinear Systems, 2nd Edition*. (Upper Saddle River, NJ: Prentice-Hall), 2002.

[8] K. Z. Haigh, S. Varadarajan, and C. Y. Tang, "Automatic learning-based MANET cross-layer parameter configuration," in *Workshop on Wireless Ad hoc and Sensor Networks (WWASN2006)*, Lisbon, Portugal, 2006, to appear.

[9] J. Bicket, "Bit-rate selection in wireless networks," Master's thesis, MIT, Feb 2005. [Online]. Available: http://pdos.lcs.mit.edu/papers/jbicket-ms.pdf

[10] D. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, pp. 134–146, Sept. 2003.