Distributed Decision Making in Cognitive Radio Networks Through Argumentation

Brent Horine, Ladislau Bölöni, and Damla Turgut
School of Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816-2362
{bhorine,lboloni,turgut}@eecs.ucf.edu

***Abstract*—We have developed a multi-agent negotiation system to distribute decision making in cognitive radio networks through argumentation. The challenge in wireless network negotiation is to efficiently exchange information to facilitate a deal without incurring excessive communication overhead or indeterminate negotiation time. Our goal is to improve both total network throughput and the number of total supported connections. We detail a set of rules, a protocol, and a compact set of messages to conduct these negotiations and complete in finite time and with little overhead. We describe our simulation environment and present results of an illustrative scenario with various conditions. This scenario includes the ability of an agent to assert high priority, possibly triggering a downgrade of an existing, non-priority connection to a slower rate in order to accommodate more connections. We compare our system's total network throughput, number of connections, and request satisfaction score to several baselines with various levels of reconsideration and conclude that our system outperforms these other approaches in all metrics.**

I. Introduction

Ad hoc radio networks offer the advantages of providing robust communications without the need for infrastructure, and in particular, centralized control. To fully take advantage of these systems, some distributed means of admitting individual communications onto the media (i.e. spectrum) is required, a process known as session admission control (SAC). Since radio spectrum is, for practical purposes, a finite resource and only a certain data rate can be supported in that spectrum under given radio conditions, at some level of network loading, the radio nodes will be unable to support any additional communication links. We seek to extend this boundary through negotiations.

The distributed decision-making process is effectively modeled using a multi-agent framework. Each radio node acts as an independent agent that cooperates with other agents to accomplish a goal. In this study, these agents are not individually rational, but instead seek to maximize the satisfaction of lease requests, total network throughput, and total number of connections.

We consider the problem of allocating link capacity for long duration streaming media of three discrete grades: high definition video (HDV), standard definition video (VID), and audio (AUD). The links between source and destination are capacity limited at high network loads, causing requests for new leases to be rejected. We are interested in ways to recover from these rejections. This can be accomplished by attempting alternate routes, or arguing that the requested connection has a higher priority than one or more existing connections (granted by some higher, agreed-upon authority). It is possible for an ongoing connection to be downgraded from a higher level to a lower one under certain conditions, although no connection will be dropped in favor of a new one.

In [6], we used argumentation to negotiate links in a simple topology using both priority and fairness arguments along with information exchanges about entities that are only known to some agents. This work demonstrated that argumentation can lead to superior solutions compared to simple negotiation. Our work here uses a simpler negotiation system, but on a larger scale with more nodes and multiple requests issued over a simulated 8 hours.

II. Preliminaries: Rules and Messages

The negotiations are governed by a set of rules, some explicit and others implicit. These rules determine acceptance of proposals and arguments. They also determine which actions to take at any given point in the negotiations, for example, in response to a received message. This begins to define the negotiation protocol. We develop it more explicitly in the next sections.

A. Implicit Rules

The nodes trust other nodes' arguments, such as priority. Nodes are conservative in that they do not ask for a higher grade of service than they need. Relay nodes do not discriminate between their sourced or sinked connections and a relayed connection. Agents know the capacity and actively maintain knowledge of the current load of their one-hop routes.

B. Explicit Rules

A node shall attempt to create a connection along the shortest route. Each node along the path shall accept the proposal if it has enough remaining capacity to support the requested grade of service. Otherwise, it will reject it. Received rejections result in an identical request, but along the next shortest route. A priority connection can force a current, non-priority connection to downgrade from HDV to VID in order to free up capacity to support the new priority connection. Conversely, a node shall accept a downgrade proposal when presented with a priority argument, unless it can assert its own priority for the specified connection.

Because of the plurality of routes in a densely connected network and the additional considerations due to priority arguments, many solutions may be possible at any given negotiation step. In highly dynamic mobile radio networks, possibly using dynamic spectrum access, lengthy negotiations may not be practical. The limits are application dependent. This work uses fairly severe limitations on the number of negotiation options to be explored by any agent. In particular, if a given route includes an over-capacity link, another route will be chosen (if available), but if that one is also over-capacity, no more routes will be explored. If the desired connection has a high priority status, a single attempt will be launched from the originator, before failing. Likewise, when a priority argument is received after rejecting a proposal, the node will search for a connection to downgrade in order to clear capacity for the priority connection. The downgrade proposal will then be sent to the source node for that connection. It is possible that the connection will also have a high priority, in which case the proposal will be rejected. Only a single attempt is made for each argument. This approach avoids the danger of consuming excessive network resources in negotiation at the expense of a higher failure rate. Obviously, some applications will call for more diligent negotiation.

C. Messages

The various radio agents or nodes communicate with each other via a small number of message types. Because communication overhead should be minimized in capacity constrained systems, these messages are very compact.

- **Propose:** requests a lease to communicate from a source node to a destination at a specific grade of communication. It specifies a start and stop time. It also includes a route.
- **Accept:** indicates that the proposal is acceptable and flows back through the reversed route to the proposal originator.
- **Reject:** issued when a node is unable to support the proposal due to a capacity constraint. It includes an argument specifying the node that failed the capacity constraint. This information can be used by the proposer to synthesize a counter proposal.
- **Argue:** sends a fact supporting a proposal. A priority assertion takes on unknown, normal, or high priority status. High priority statuses are associated with particular connections and can force a downgrade of another connection in order to support the priority one.
- **Downgrade:** specialized proposal sent when a relay does not have the capacity to support a proposal, but has received a priority argument. The downgrade is sent to the originator of a downgrade-able connection.
- **Notify:** informative message that spreads the news of an accepted downgrade to other nodes in the connection route that is not on the path of the original downgrade and accept message pair.
- **Confirm:** sent by the ultimate destination of the notify message back to the originator of the notify. This is primarily used to synchronize message passing so that node that initiated the downgrade proposal can either accept the original proposal (if it is the final destination) or forward it.

III. Proposed Protocol

The rules and messages combine to a set of algorithms that represents the negotiation protocol. Negotiations start when a node has a desire to communicate with another at a specific rate. If appropriate, a high priority status is attached to the desire. This becomes an intention with a start time, duration, and a route. A proposal in the form of a request is then created and dispatched to the first node in the route.

Requests or proposals are accepted if sufficient excess capacity exists. This process is described in Algorithm 1. The agent examines the outbound connection at each node and compares the current load plus the proposed load to the capacity limit. If it is below the limit, the proposal is forwarded. Notice that if the current node is the final destination, the proposal can be immediately accepted, since the capacity has already been verified by preceding nodes. Actual acceptance from intermediate nodes is deferred until the destination nodes have accepted the proposal. When an intermediate agent forwards a proposal, it reserves the capacity on the assumption that the connection will ultimately be accepted by all nodes along the route. Eventually, the node will receive a response message from the node to which it forwarded the proposal. If it is an acceptance, then the reservation becomes a commitment for the duration of the connection. If it is a rejection, the reservation is cancelled. In either case, the message is forwarded along the reverse route back to the originating node. Timeouts can be used to clean up the negotiation state in case of a broken link. When an initial proposal is rejected, the originating agent can either issue a proposal along a different route, or issue an argument, if one is available, as shown in Algorithm 2.

If an agent receives an argument, it retrieves the cached proposal to which it refers (Algorithm 3). In the case of a priority argument, it attempts to find an existing connection that it can downgrade in order to clear capacity for the proposed connection. Assuming it finds one, it then issues a downgrade proposal to the originator of that connection. This gives that agent an opportunity to issue its own argument, for example, it also has priority, against the downgrade. Otherwise, it accepts it. Agents handle a downgrade request by forwarding to the originating agent of the connection to be downgraded, which is the destination of the message route. The agent retrieves the connection from its database and checks to see if it has a high priority status. If it does, it rejects the downgrade request; otherwise, it accepts it and makes the appropriate changes to its connections and link loading databases. In either case, the message is sent back along the reverse route. Acceptances from both proposals and downgrade requests require the updating of the connections and links databases.

If the node issuing the downgrade proposal is an intermediate node in the downgraded connection's route, it needs to notify the other agents in the opposite direction from the originator. This process is accomplished through a notification message and a corresponding confirmation. No decisions are required at any of the nodes in this message route. As before, actual changes to the connections and link loading databases are deferred until the confirmation message is received.

Algorithm 1 Handle Proposal algorithm

```
function HANDLEPROPOSAL(Proposal proposal)
    if isFinalDestination() then
        $new\_msg \leftarrow accept(proposal)$
        connections.add(new Connection(proposal))
    else
        if $current\_load + proposed\_load \leq maximum\_capacity$ then
            $new\_msg \leftarrow forwardMessage(proposal)$
        else
            $new\_msg \leftarrow reject(me, proposal)$
        end if
    end if
    return $new\_msg$
end function
```

Algorithm 2 Handle Reject algorithm

```
function HANDLEREJECT(Reject reject)
    if isOriginatingNode() then
        if $num\_attempts = 1$ then
            $route\_num \leftarrow route\_num + 1$
            $new\_msg \leftarrow createProposal(route\_num)$
            $num\_attempts \leftarrow num\_attempts + 1$
        else
            if havePriority(desire) then
                $new\_msg \leftarrow createArgument(priority)$
            else
                Fail                    ▷ Negotiation fails and terminates
                $new\_msg \leftarrow null$
            end if
        end if
    else
        $new\_msg \leftarrow forwardRejection(reject)$
    end if
    return $new\_msg$
end function
```

Algorithm 3 Handle Argument algorithm

```
function HANDLEARGUMENT(Argument arg)
    $fact \leftarrow arg.fact$
    if priority then
        $connection \leftarrow findDowngradeConnection$
        if $connection \neq null$ then
            $new\_msg \leftarrow createDowngrade(connection)$
        else
            $new\_msg \leftarrow reject$
        end if
    end if
    return $new\_msg$
end function
```

IV. Simulation Study

We developed a simulator to compare the performance of the negotiation technique to a variety of loads. We first discuss the metrics that guide the design of the simulator system, which we review before presenting the results of an illustrative scenario.

A. Metrics

The performance measures include the overall network throughput and the satisfaction rate of requests under various loading conditions. We also investigate the satisfaction of individual links in the context of downgrades during the connection lifetime due to priority or fairness assertions.

An additional and very significant measure in negotiation is the number and total size of messages used to complete the negotiations. The number of messages can be minimized by passing all of the data in the knowledgebase in each message. On the other hand, this can increase the total load, especially when no deal is possible or a simple deal is possible. In this case, the transmission of the extra data is either futile or unneeded. With a small network load, it is likely that initial proposals will be immediately accepted and there is no need to transmit excessive information. With very heavy loads, it becomes more unlikely that a deal can be reached; the network is at full capacity. Of course, the node may still wish to engage in negotiation and should send its strongest argument. It is reasonable for the originating node to consider its own knowledge of the network load and adapt its negotiating strategy accordingly. In the current work, we choose to transmit less information per message. This approach allows us to see the progress of decisions in the negotiation process more clearly, since the result of each significant decision is a message. While we measure the number of messages required to complete a negotiation, one should be cautious about drawing final decisions based upon this information.

The overall achieved performance is characterized in several ways. The total carried network load, or throughput is found by summing the grade for each unique connection. This can be further analyzed by comparing it to the requested capacity. This will account for the impact of downgrades. Similarly, satisfaction measures how good of a deal is reached for a request, with zero signifying a complete rejection, one signifying acceptance of the original request, and a number in (0,1) reflecting the degree of satisfaction when the achieved grade is less than the requested grade. The total number of unique connections measures the ability of the system to support as many users as possible. Finally, the number of hops it takes to support the connections measures the impact of rerouting to non-optimal routes.

B. Simulator Design

With the metrics, rules, and messages defined, we now discuss the design of the simulator system. Each of the messages described in section II inherit from a common class. This class encapsulates the routing data common to all messages. A topology class controls the generation of a specified number of nodes, randomly determines the connectivity between each pair of nodes, and calculates a number of routes between each pair. This simulation assumes homogeneous links with a specified maximum capacity, although nothing in the protocol precludes heterogeneous links. Using identical links with either no capacity or $MAX\_CAPACITY$, the results are more clearly discerned. Options are available to study both a densely and a sparsely connected network, but the settings are adjusted to ensure that all nodes can be reached from any node. In other words, these are all topological spaces. Because of this, failures can be attributed to a load versus capacity constraint rather than a disconnected graph.

A scenario generator issues desires based upon a Poisson distribution, parameterized by its mean, λ. It is called once per simulated second. This results in a mean request rate of λ requests per simulated second. It creates an intention from this desire by randomly choosing an originating node and a distinct destination node according to a uniform distribution. It enforces the distinct condition by repeatedly drawing a node number from the distribution until it is different than the originating node. The requested connection grade is chosen from a uniform distribution, $[0, 1]$, such that if the drawn number is in $[0.67, 1.0]$, a HDV connection is requested. A VID connection is used if the number drawn is in $[0.33, 0.67)$. Otherwise, an AUD connection is requested. A high priority status is set for the connection if a uniform distribution generator provides a value in $[0.0, 0.75)$. Finally, the start time of the connection is set to the current time (assuming that the negotiation time is negligible) and the stop time is set such that the duration is drawn from a Poisson distribution with mean of 3600, i.e. one hour.

If an intention is generated at a particular time step, it is set in the originating node and that node subsequently attempts to generate a proposal based upon that intention. The proposal generation process can fail if there are no outbound links with sufficient capacity, and in the case of the argumentation agent, no argument can be formed. This is logged as a failure. A message queue dispatches messages to the target agents message handling routine.

At each time step, each node cleans up any expired connections. Then the scenario generator is called. If one or more intention is generated, the negotiations are conducted. Finally, the results are scored. Scoring involves retrieving a set of distinct connections from the agents. The grade of each connection is summed to calculate the total network load, along with counting the number of connections to assess how many users are supported.

C. Illustrative Scenarios

The simulator allows us to investigate a number of scenarios using different agent models. As a baseline, we developed a radio agent (RadioNode) that simply requests the shortest path. If the capacity is available, it is accepted, otherwise it is rejected. A slightly more sophisticated model (RadioNodeReRoute) will attempt the next longest route upon receiving a rejection. Our RadioNodeABN conducts negotiations according to the rules we have discussed in section II. Finally, we also implemented an oracle (RadioNodeOracle) that has full knowledge of the link loadings throughout the network. When proposing, it is able to choose a route that it knows will succeed, if one exists. On the other hand, it allocates on the fly, temporally. It also chooses the shortest available route, rather than another possible option, the least loaded route. Most importantly, it does not reconsider existing routes when analysing current requests. When considered within the context of the degrees of freedom associated with our rules, an optimum approach is ill-defined. Our rules favor supporting as many connections as possible, while respecting a possible priority condition. This is different than maximizing total data throughput in the network or conventional fairness criteria.

TABLE I
SIMULATION PARAMETERS

Parameter	Min	Max
Number of nodes	7	
Capacity	48 Mbps	
Lambda	0.0001	0.05
Request rate	0.36/Hr	180/Hr
Connection density	sparse	dense

We consider three connections grades, high definition video (HDV) at 8Mbps, regular video (VID) at 3.5 Mbps, and audio (AUD) at 256 kbps. Table I lists the remaining simulation parameters.

The exchange of one of the more complicated negotiations at time step 23347 (6:29:07) when $\lambda = 0.017$ (61.2 requests/sec.) is listed in Table II and diagrammed in Fig.1. The initial proposal is for VID from node F to node E. It is rejected along the route by node C. This information is passed back to node F, which then finds a new route from F to E that does not include C. This route also fails, this time at node A. Node F then asserts a high priority status through an argument targeting node A. Node A finds a connection from D to B that passes through A that can be downgraded from HDV to VID to clear up room for the proposed connection. It then sends a downgrade request to node D, which accepts the request. Since node A is just an intermediate hop on the downgraded connection's route, it sends a notification in the opposite direction to the destination node, i.e. towards B. Once A receives the confirmation associated with the notification, it again retrieves the original proposal to forward to the next node on the route. Finally, accept messages flow back to node F and the negotiation concludes successfully, at the expense to the connection between D and B of a single level downgrade. The confirmation message actually arose in order to keep the simulation synchronized in terms of message passing. It does seem useful in an implementation in order to roll back the transaction if a link is broken somewhere. The protocol as outlined here can be made transactional to a point if reliable message handling is used and a rollback message added. This would be necessary to keep the agents' knowledgebases consistent in terms of active connections and link loadings. Rollback of a downgraded connection would be the greatest challenge, mostly complicated by the record keeping necessary to perform a rollback.

TABLE II
MESSAGES PASSED IN SCENARIO 1

#	S	R	Message
			$Desire, F, E, VID, 23347, 26965$
1	F	E	$Request, F, E, VID, 23347, 26965, [A, F, C, E]$
2	E	F	$Reject, C, NOCAPACITY$
3	F	A	$Request, F, E, VID, 23347, 26965, [F, A, D, E]$
4	A	F	$Reject, C, NOCAPACITY$
5	F	A	$Argue, HIGHPRIORITY$
6	A	D	$Downgrade, Conn[D, B], HDV, VID, [A, D]$
7	D	A	$Accept, [D, A]$
8	A	B	$Notify, HDV, VID, Conn[D, B], [A, B]$
9	B	A	$Confirm, [B, A]$
10	C	E	$(fwd)Rq, F, E, VID, 23347, 26965, [F, A, D, E]$
11	D	E	$(fwd)Rq, F, E, VID, 23347, 26965, [F, A, D, E]$
12	E	D	$Accept, [E, D, A, F]$
13	E	D	$Accept, [E, D, A, F]$
14	A	F	$Accept, [E, D, A, F]$

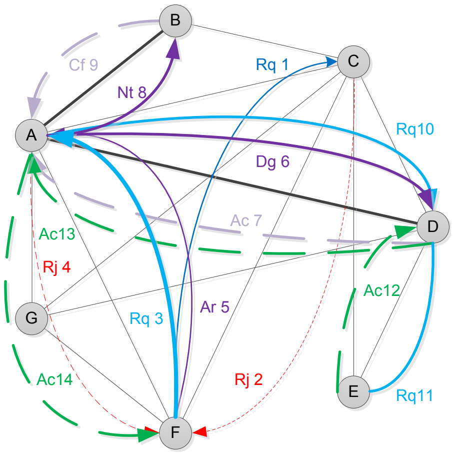

Fig. 1. Message exchange including downgrade and notify messages.

D. Results

The top chart in Fig.2 demonstrates the superior total bits transferred over a simulated day for a densely connected network at a high request rate where it is challenged in terms of load. Our argumentation based approach yield a clearly higher throughput.

We characterized the number of active connections supported over time. The scenario generator randomly creates desires for connections at one of three different grades with durations around a mean of one hour. With a review of the middle chart in Fig.2, one can see that the argumentation approach supports more connections than any of the other techniques at higher request rates. This is accomplished mainly through the downgrade process that is essential to satisfying new priority connections, while preserving existing connections' ability to communicate, albeit at a low rate.

The bottom chart in Fig.2 illustrates the failure rate for the various models at a request rate of 180 requests/hour with both sparse and dense connectivity. As expected, the argumentation approach has the lowest failure rate. This is due to the ability to downgrade connections in order to squeeze more into the finite capacity. Densely connected networks suffer fewer failures

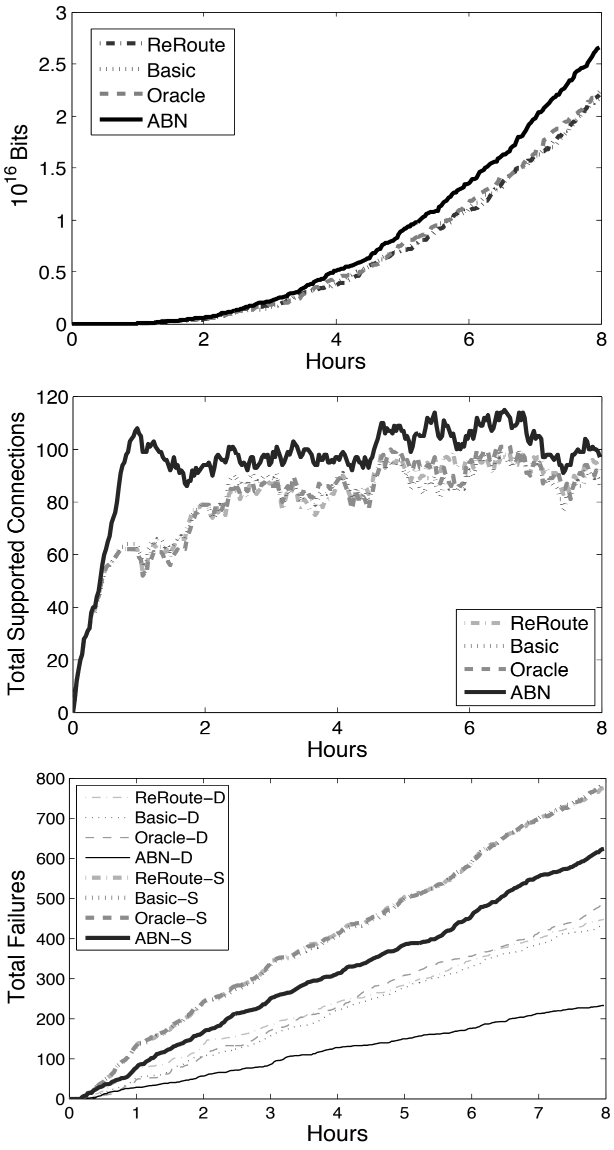

Fig. 2. Top:Total bits transferred with high request rate densely connected, middle: Total active connections with high request rate sparsely connected, bottom: Failures with high request rate

than sparsely connected ones since there are more options to explore. It is interesting to note that the ReRouting agent performs worse than the basic agent. Even though re-routing may cause a single proposal to be accepted, it actually loads more nodes, thereby making future proposals more likely to run into capacity constraints. Perhaps in actual operation, the greater number of hops might mean a shorter distance between each hop. This would enable a higher order modulation to maintain a constant signal to noise ratio at the same power, leading to a narrower bandwidth. Ultimately, the channel could support more connections. Our model does not account for these details. Since the Oracle model has perfect knowledge of the link capacities throughout the network, it never fails during a negotiation. Instead, at high loads, it fails at the very beginning in trying to create a proposal from the intention,

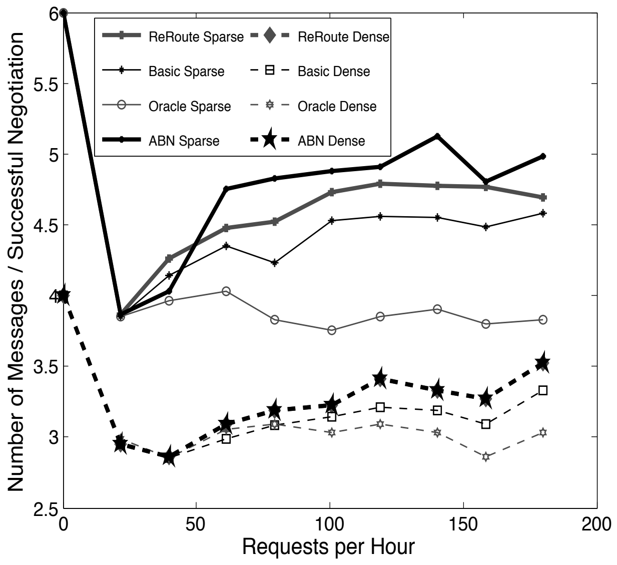

Fig. 3. Number of messages per successful negotiation.

because it cannot find an available route.

One of the concerns in negotiations is the overhead and latency induced by the need to negotiate. We estimate this by examining the number of messages required for each negotiation model. As seen in Fig.3, there is only a small penalty in using the more powerful argumentation approach. Recalling our statements in metrics subsection, these predictions are likely to be pessimistic compared to an operational system.

V. Related Work

A great deal of research has focused on the challenge of guaranteeing a specified quality of service (QoS) in mobile ad hoc networks (MANET). Some of these proposed solutions work with scheduled media access control (MAC) layers, while others work with contention based MACs. One of the challenges of these systems is having to deal with local information, keeping exchanged information fresh, and estimating channel capacity conditions [5]. Pitt et al. [11] use a norm-governed multi-agent system to address QoS provisioning in MANETs. Their system runs over a Multimedia Network Support Platform (MNSP). The MNSP provides the communication services while the multi-agent system provides the decision making services through a deliberative process according to norm-governed policies and protocols. Förster surveys machine learining techniques in wireless ad hoc networks in [3], covering reinforcement learning, swarm intelligence, heuristics, and mobile agent technologies. Liu and Issarny study trust relationships through reputation for MANETs in [7]. Finally, Gan et al. address energy efficiency using agent techniques in [4].

Rahwan et al. present the limitations of game theoretic and heuristic approaches to negotiation and how argumentation can overcome them [12]. A number of researchers have explored argumentation based negotiation as a means of changing preference relationships during negotiations [1], negotiating with constraints [8], and deciding with incomplete, uncertain, or inconsistent information [9], [15]. A number of frameworks have been proposed [2], [12], [13], [14] and been analyzed formally [1] and empiracally [10].

VI. Conclusions

We have presented a negotiation protocol to allocate leases for radio connection in an ad hoc network. The process is entirely distributed and is governed by a simple set of explicit and implicit rules. A well defined set of messages exchange requests, results, and arguments between nodes. Simulations indicate that our system produces superior results in terms of maximizing accepted connections. The system is reliable under a variety of request rates and with both a sparsely and densely connected network.

References

[1] L. Amgoud and S. Vesic. A formal analysis of the role of argumentation in negotiation dialogues. *Journal of Logic and Computation*, 2011.

[2] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and -person games. *Aritificial Intelligence*, 77(2):321–357, September 1995.

[3] A. Förster. Machine learning techniques applied to wireless networks: Guide and survey. In *3rd International Conference on Intelligent Sensors, Sensor Networks and Information*, 2007.

[4] L. Gan, J. Liu, and X. Jin. Agent-based, energy efficient routing in sensor networks. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, 2004.

[5] L. Hanzo II and R. Tafazolli. A survey of QoS routing solutions for mobile ad hoc networks. *IEEE Communications Surveys & Tutorials*, 9:50–70, 2007.

[6] B. Horine, L. Bölöni, and D. Turgut. Argumentation based negotiation in cognitive radio networks. In *The 37th IEEE Conference on Local Computer Networks*, 2012.

[7] J. Liu and V. Issarny. Enhanced reputation mechanism for mobile ad hoc networks. In *Second International Conference on Trust Management: iTrust*, 2004.

[8] M. Mbarki, J. Bentahar, B. Moulin, and A. Moazin. Constraints-based negotiation using argumentation. In *Proceedings of the 13th International Workshop on Non-Monotonic Reasoning (NMR)*, 2010.

[9] S. Modgil, F. Toni, F. Bex, I. Bratko, C. I. Chesñevar, W. Dvořák, M. A. Falappa, X. Fan, S. A. Gaggl, A. J. García, et al. The added value of argumentation. In *Agreement Technologies*, pages 357–403. Springer, 2013.

[10] P. Pasquier, R. Hollonds, I. Rahwan, F. Dignum, and L. Sonenberg. An empirical study of interest-based negotiation. *Autonomous Agents and Multi-Agent Systems*, 22(2):249–288, 2011.

[11] J. Pitt, P. Venkataram, and A. Mamdani. QoS management in MANETs using norm-governed agent societies. In O. Dikenelli, M.-P. Gleizes, and A. Ricci, editors, *Engineering Societies in the Agents World VI*, volume 3963 of *Lecture Notes in Computer Science*, pages 221–240. Springer Berlin / Heidelberg, 2006.

[12] I. Rahwan, S. D. Ramchurn, N. R. Jennings, P. McBurney, S. Parsons, and L. Sonenberg. Argumentation-based negotiation. In *The Knowledge Engineering Review*, volume 18, pages 343–375. Cambridge University Press, Dec 2003.

[13] C. Sierra, N. R. Jennings, P. Noriega, and S. Parsons. A framework for argumentation-based negotiation. In M. P. Singh, A. Rao, and M. J. Wooldridge, editors, *Intelligent Agents IV: Agent Theories, Architectures, and Languages: 4th*, volume 1365 of *Lecture Notes in Artificial Intelligence*, pages 117–192. Springer-Verlag, July 1997.

[14] N. Turan, T. Dai, K. Sycara, and L. Weingart. Toward a unified negotiation framework: Leveraging strengths in behavioral and computational communities. In *Models for Intercultural Collaboration and Negotiation*, pages 53–65. Springer, 2013.

[15] W. Visser, K. V. Hindriks, and C. M. Jonker. Argumentation-based qualitative preference modelling with incomplete and uncertain information. *Group Decision and Negotiation*, 21(1):99–127, 2012.