Using Organization Knowledge to Improve Routing Performance in Wireless Multi-Agent Networks

Huzaifa Zafar, Victor Lesser, Daniel Corkill, Deepak Ganesan
Department of Computer Science
University of Massachusetts Amherst
{hzafar, lesser, corkill, ganesan}@cs.umass.edu

ABSTRACT

Multi-agent systems benefit greatly from an organization design that guides agents in determining when to communicate, how often, with whom, with what priority, and so on. However, this same organization knowledge is not utilized by general-purpose wireless network routing algorithms normally used to support agent communication. We show that incorporating organization knowledge (otherwise available only to the application layer) in the network-layer routing algorithm increases bandwidth available at the application layer by as much as 35 percent. This increased bandwidth is especially important in communication-intensive application settings, such as agent-based sensor networks, where node failures and link dynamics make providing sufficient inter-agent communication especially challenging.

Categories and Subject Descriptors

C.2.2 [**Network Protocols**]: Routing Protocols; C.2.4 [**Distributed Systems**]: Distributed Applications; I.2.11 [**Distributed Artificial Intelligence**]: Multi-Agent Systems

General Terms

Algorithms, Design, Experimentation, Verification

Keywords

Agents, Multi-Agent Sensor Networks, Organization Design, Wireless Routing, Communication, Bandwidth

This work is supported in part by the AFRL "Advanced Computing Architecture" program, under contract FA8750-05-1-0039. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views contained in this paper are the authors.'
This work was also supported by the Engineering Research Centers Program of the National Science Foundation under NSF Cooperative Agreement No. EEC-0313747. Any Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.
Cite as: Using Organization Knowledge to Improve Routing Performance in Wireless Multi-Agent Networks, Zafar H., Lesser V., Corkill D., and Ganesan D., *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp. 821-828.

1. INTRODUCTION

In any multi-agent system (MAS), there is a cost in communicating among agents in the form of time delay and bandwidth expenditure. In wireless networks, agents are often multiple communication hops away from one another. Supporting multi-hop communication involves an additional network-layer exploration cost in determining communication paths to these agents. This exploration cost becomes significant as the size of the network increases (Table 1) and as network stability decreases. Exploration expenditures reduce the effective bandwidth available to the application, resulting in additional time required in transferring the same amount of application data from one agent to another.

Size of Grid	OLSR		DSDV	
	Total	Per Agent	Total	Per Agent
9 agents	1038	115.33	178	19.77
16 agents	3880	242.5	521	32.56
25 agents	9401	376.04	1011	40.44
36 agents	17535	487.08	1654	45.94
49 agents	32353	660.26	2136	43.59

Table 1: Exploration costs increase with size of the network. We use a wireless ad hoc grid network. Each agent lies on the edge of the transmission range of its neighboring agents, allowing for only 4 neighbors per agent. The table shows the number of exploration messages sent in the first 60 seconds for OLSR (a standard routing algorithm) and for DSDV (a standard proactive distance-vector-based routing protocol)

CNAS (Collaborative Network for Atmospheric Sensing) [3] is one such wireless MAS application. It is a collection of sensor agents, spread sparsely over a geographical region. CNAS makes an overall assessment of detailed local atmospheric conditions by combining data from sensor agents within predefined cluster areas. A hierarchy of communication is used, where all the sensor agents collect data from their environment and transfer it to a dynamically designated sensor agent performing a second role of a *cluster head*. The cluster head is responsible for aggregating and reporting the conditions in its cluster area. Moreover, the cluster head is also responsible for providing direction to the rest of the agents in its cluster. Due to network dynamics such as agent and link failures and changes in communication-link characteristics, the network-level routing algorithm at each agent in the network performs regular exploration of links to its neighbors. This exploration determines neighbor availabil-

ity and the utility of the links with them. Agents acting as cluster heads communicate with a *regional agent*, but with less frequency. Nevertheless, paths to the regional agent must also be maintained. However, regional-agent communications are of extremely high priority and there are greater consequences for delayed delivery.

MAS applications use organization design to determine when to communicate, how often, with whom, with what priority, and so on [4, 5]. By being aware of which agents are interested in communicating with whom at the network level, we are able to direct network-layer exploration where it is most beneficial, find optimal paths faster, and conserve significant bandwidth for application use. We also make sure of the organizational importance (priority) of various agent communication. General-purpose routing algorithms treat each message as having the same priority. These algorithms spend the same amount of energy exploring paths for application messages where it is extremely important to find the best possible path, as they do exploring paths for messages that are not as important, and can be delayed or routed less directly. By using the organizational priority of communication, we are able to improve application performance with fewer exploration messages.

We modified a pre-existing proactive routing protocol (CQRouting [8]) to incorporate the organization knowledge that specifies for each agent whom to communicate with, the priority of the message being sent, and the frequency with which the communication happens. Using a detailed network-level simulation of CNAS we obtain a large reduction in the number of exploration messages relative to traditional routing protocols. This results in a significant increase in the communication bandwidth available to the CNAS application.

2. RELATED WORK

Developing a path to a destination agent is performed by the underlying routing algorithm, and there has been considerable research done in developing general-purpose routing algorithms [1, 10, 11, 12]. However all proactive routing algorithms (where the routing table is developed before sending application-level messages) develop paths to every node in the network, irrespective of the need for or importance of the path. This results in significant discovery and maintenance cost on the part of the routing algorithm due to the assumption that every node wants to communicate with every other node in an extremely dynamic network. Moreover, routing algorithms explore more frequently as the dynamics of the network increase. On the other hand, reactive routing algorithms suffer from an added exploration delay when sending data messages (on top of the delivery delay) and are not suitable for highly responsive agent networks.

2.1 OLSR

OLSR (Optimized Link State Routing protocol) [2] is the most commonly used proactive routing protocol in wireless sensor networks. OLSR is optimized for large, dense, wireless ad-hoc networks, and computes optimal forwarding paths by flooding exploration messages to all nodes in the network. Exploration in OLSR happens in two stages. The first stage performs a distributed election of a set of multi-point relay (MPR) nodes that are solely responsible for forwarding messages within the network. The MPR set has the property that each node in the network is a direct neighbor of an MPR, and that two MPRs are at most separated by a single node. The second stage in OLSR develops a routing table at each node that defines the next hop on the path to every other node in the network. This is done by having each node flood the network with topology control (TC) messages, and using other TC messages in generating its local routing table. By using only the MPR nodes to propagate exploration messages, OLSR reduces some of the redundancy of the flooding process. OLSR uses *destination-based exploration*, where a node develops a table entry for a destination only after it receives a TC message from that destination.

2.2 CQRouting

CQRouting [8] is an extension to the QRouting [1] protocol based on the distributed QLearning algorithm [13]. The network is constructed from a distributed collection of learners. A QValue, $Q_x(y, d)$, for a node, x, is the expected time taken to transfer a message to destination node, d, through neighbor node, y. A policy determines which neighbor a message is forwarded to so that it reaches its destination with minimum delay.

QRouting uses the "Full Echo" algorithm to perform exploration [1]. Each node in the network periodically requests the policy being used by each of its neighbors in determining paths to various nodes in the network (which is determined by the routing table of that neighbor, generated using the Bellman-Ford Algorithm for routing, first described in the ARPANET [9]). The querying node then updates its own tables based upon the following QLearning equation:

$$Q_x(y, d) = Q_x(y, d) + \alpha(Q_y(\hat{z}, d) + q_y - Q_x(y, d)) \qquad (1)$$

where α is the learning rate that regulates the effect of previous QValues on the current one. $Q_y(\hat{z}, d)$ is the delay in sending a message from node y, through y's best neighbor $\hat{z}$, to node d. q_y is the delay in sending a message from node x to node y.

In CQRouting each node also maintains a confidence in its QValues, and shares this confidence with other nodes in the network along with its routing tables. The closer the confidence value is to 1, the better the node expects the QValue to reflect the state of the network. When an agent updates its QValues from the policies received from its neighbors, the agent uses the confidence its neighbor reports in determining its own QValue as follows

$$Q_x(y, d) = Q_x(y, d) + \alpha C_x(y, d)(Q_y(\hat{z}, d) + q_y - Q_x(y, d)) \quad (2)$$

For each time step that the agent does not explore, its confidence decays due to the expectation that the network is dynamic and its QValues might not reflect the current state of the network.

3. eCQROUTING

We develop a new wireless routing protocol called eCQRouting, that combines and extends OLSR and CQRouting. OLSR pays a higher exploration penalty due to the communication required in setting up the MPR nodes and in developing nodes' routing tables. In eCQRouting, organization knowledge is used to determine the explorer nodes. The rest of the nodes in the network take advantage of these exploration messages in developing their own routing tables by eavesdropping on the exploration messages and their results. Secondly, OLSR uses frequent network flooding to keep the

routing tables synchronized with fluctuations in the network. Incorporating CQRouting with OLSR smooths the effect of network fluctuations, reducing the need for frequent flooding of exploration messages. Furthermore, by reducing flooding, the benefit of electing MPRs is also reduced. This reduction is enough that it does not justify the election cost of MPRs, so we remove MPR development from eCQRouting. The following subsections describe additions to the eCQRouting protocol to allow for a smoother implementation in wireless sensor networks.

3.1 Damping Factor

Use of a learning factor (α) in QLearning, introduces the possibility of the utility of agent A dropping below the utility of agent B, even though B is being used as its next hop. Since we are interested in minimizing QValues, whenever A sends its policy to B, B will switch to using A as its next hop due to the lower utility. By performing multiple explorations over that path, QLearning eventually overcomes this issue and converges to the corrects QValues. However, because every exploration message consumes potential application-level bandwidth, we modified the QRouting algorithm to dynamically adjust the learning factor each time an agent updates its QValues. The new learning factor depends on the current QValue and the QValue provided by the agent's neighbor. This allows for better and faster convergence of QValues with fewer explorations while retaining the benefit of the learning factor in keeping the optimal path from flip-flopping between two close paths. The learning factor regulates how closely the current QValue reflects the "present" delay to the destination verses the values calculated in the "past." This implies that the closer α is to being equal to one, the stronger the guarantee that the QValue of any agent to its destination is greater than the QValue of its next hop to the destination. From the QLearning algorithm:

$$\begin{aligned} Q_x(y,d) + \alpha(Q_y(\hat{z},d) + q_y - Q_x(y,d)) \quad &> \quad Q_y(\hat{z},d) \\ \alpha > \frac{Q_y(\hat{z},d) - Q_x(y,d)}{Q_y(\hat{z},d) + q_y - Q_x(y,d)} \end{aligned} \tag{3}$$

The above value gives us a lower bound on the value of α, with the upper bound being 1. The new damping factor is:

$$\alpha = 1 - \hat{\alpha} * (1 - \frac{Q_y(\hat{z},d) - Q_x(y,d)}{Q_y(\hat{z},d) + q_y - Q_x(y,d)}) \tag{4}$$

By changing $\hat{\alpha}$, we are able to regulate how close we want our α value to be to 1, but at the same time avoid damping issues. For eCQRouting, we modified the α formula as follows

$$\alpha = 1 - \hat{\alpha} * (1 - \frac{Q_y(\hat{z},d) - Q_x(y,d)}{C_x(y,d) * (Q_y(\hat{z},d) + q_y - Q_x(y,d))}) \tag{5}$$

4. ORGANIZATION AND ROUTING

We next describe incorporating organization knowledge in the routing algorithm used at each agent and the corresponding routing-level exploration changes, given this knowledge.

4.1 Knowledge of the organization

The direction and priority of communication between each agent role is represented as a weighted graph. A local copy of the graph is to used to make exploration decisions based on the roles an agent is assigned. Figure 1a shows the CNAS organization. The corresponding communication graph is depicted in Figure 1b.

Figure 2a shows a second, more complicated organization as used in CASA (Collaborative Adaptive Sensing of the Atmosphere) [7] another agent-based atmospheric sensor network with considerably higher bandwidth requirements than CNAS. In CASA, there are four roles an agent can take; Rad(s) (Radar(s)), FD (Feature Detector), FR (Feature Repository) and Opt (Optimizer). The organization is hierarchical. A CASA agent is responsible for communicating with other neighboring agents that are performing the same role as itself as well as communicating with parent agents, however with a lower priority. Also, agents with roles higher on the hierarchy communicate less frequently, and these messages have a much higher priority than communications among agents that are lower in the hierarchy. The corresponding graph communication graph with priorities is shown in Figure 2b.

Each agent is also provided with a description of all the other agents in the network (that are part of the organization), the roles they play, and the destination agents they communicate with. We believe that providing this kind of "global" view of the organization knowledge is reasonable, as it is considerably more stable than the routing tables used by the network-layer routing algorithms and this global perspective has negligible costs when compared to sharing routing tables. As the size of the organization increases, the scale of both the organization structure and the routing table increases. However, the organizational knowledge limits the size of the routing table by guiding exploration messages (and consecutively path development) to only those destination agents reflected in the organization knowledge.

4.1.1 Exploration Based on Organization

A destination agent in the organization knows (from its organizational knowledge) all the agents that communicate with it. Also, in most networks like CNAS, the number of destination agents is much smaller than the number of source agents. In eCQRouting, the destination agent performs periodic exploration of the network[1]. The extent and direction of an agent's exploration is based on the placement of agents that communicate with it. The initial exploration messages are used by the destination agent to find the other agents. Future exploration messages are then directed according to the location of agents that communicate with the destination. The QValue at the source agent is defined by the delay in receiving the exploration messages.

4.1.2 Exploration based on Confidences in QValues

In eCQRouting, an agent explores when its confidence drops below a certain threshold. Since exploration happens at the destination, yet confidence is calculated at the source, confidence is piggybacked on application messages from the source to its destination. The destination agent updates a local copy of the confidences of all agents that send messages to it and uses its local copy to determine when to explore. Furthermore the destination agent confidences decay over time so it will explore even if there are no application messages sent to it.

[1]We assume communication links between nodes are bi-directional and symmetrical. Even though this is not true for most wireless sensor networks, both OLSR and CQRouting assume the same.

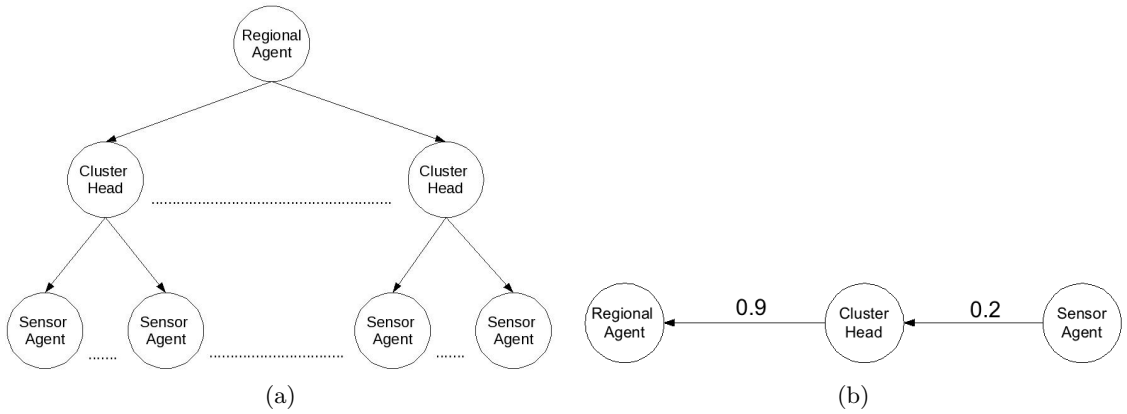


Figure 1: (a) CNAS Organizational Design and (b) CNAS communication directions and priorities

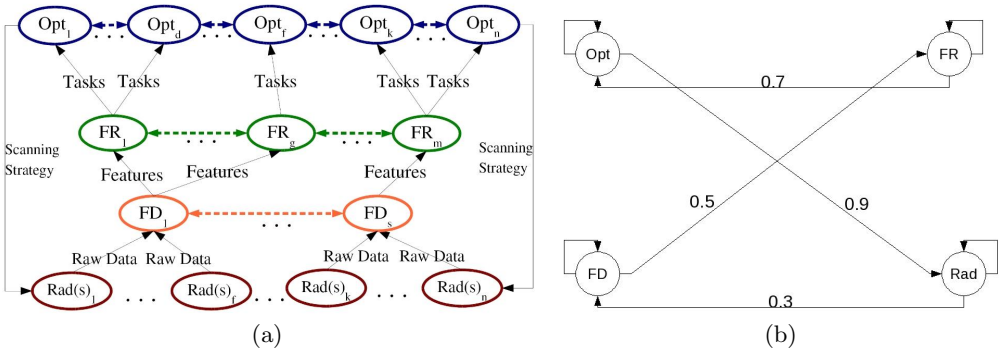


Figure 2: (a) CASA Organizational Design and (b) CASA communication directions and priorities

4.2 Message Priority

In Section 4.1, we defined our organization knowledge as a weighted graph, where the weight on an edge of the graph is based on the priority of the messages sent from one agent to another. Messages that are critical have a much higher priority as compared to other messages and require more exploration in order to maintain the optimal path, even though the number of messages sent might be much lower. For example, the "Optimizer" role in CASA includes setting the scanning strategy for the "Radar" agents. Lost scanning strategy messages can be very detrimental to the application as it could potentially mean radar agents are following an out-dated scanning strategy. CASA uses focused radars, so this could lead to loss of tracking data.

At the application level, an "Optimizer" agent can delay sending messages to other "Optimizer" agents if it has a scanning strategy that needs to be communicated. Each agent uses a priority queue which regulates when to send what message based on its importance. A problem with this strategy is priority-based starvation, where low priority messages wait infinitely while high priority messages are added ahead of them in the queue. As a routing algorithm however, we would like lower priority messages to be communicated using suboptimal paths and be eventually delivered, while saving optimal paths for high priority messages.

In eCQRouting we solve this problem by using two techniques. The first technique uses an approximate measure of message priority for each source-destination pair. The priority is then used to regulate the frequency with which exploration is performed by the destination agent. In the second technique, each agent has a local measure of the performance of the MAS based on the probability of dropping a message to the destination. For each message, decisions are then taken based on this local measure.

4.2.1 *Exploration based on message priority*

Consider a simple case shown in Figure 3. After the first

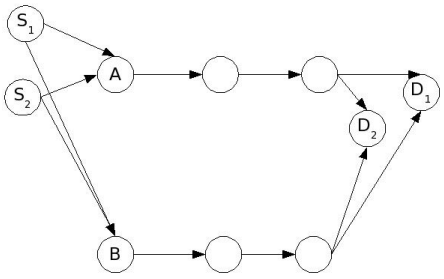


Figure 3: Priority behavior in a simple network. $S_1 \rightarrow D_1$ has a much higher communication priority than $S_2 \rightarrow D_2$

round of exploration is completed, S_1 and S_2 both deter-

mine A to be the best path from themselves to D_1 and D_2 respectively. However as both of them start using A as their next hop, the network discovers the bandwidth from A to D_1 and D_2 is insufficient to transmit both sets of messages, resulting in an increased time delay. The next time an exploration happens, both D_1 and D_2 explore, find the path through B to be the better next hop and S_1 and S_2 both choose B causing an overload on B.

In eCQRouting however, we lower the threshold at which a destination agent explores based on the priority using the formula $threshold = priority * sd\text{-}threshold$. The sd-threshold is uniform for all source-destination pairs. This tolerates a lower variance on the value of high priority messages before forcing exploration to resolve them, and results in better mean and variance development for those values. It also means that if S_1 has a higher priority over S_2, it will explore more, forcing S_1 to pick a suboptimal path through B, in order to make sure that S_1 resolves the bottleneck. The next time S_2 explores, it would keep the optimal path through A. The technique is suboptimal is a static network, but in a dynamic network permits a high priority source-destination pair to be more aware of the state of the network in developing paths to the destination.

The disadvantage of this algorithm is the system has to wait for the next exploration message to determine new paths, which delay decisions.

4.2.2 Exploration based on message loss

The second technique takes into account the expected effect of performance relative to message loss and performs local decisions based on that effect. Suppose S_1 and S_2 start sending messages and realize 10% of their messages are being dropped before they reach A. If the performance curve (see Figure 10a) for messages from S_1 is such that S_1 cannot handle the 10% loss, S_1 will pick the path through B with the expectation that that path will deliver more than 90% of its messages.

As agents explore, link probabilities are shared in the following way; assume that the network in Figure 3 has the probabilities shown in Figure 4. Here, E will share

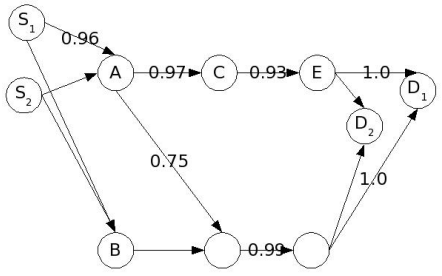


Figure 4: A simple network with probability of sending a message from one agent to another

1.0 with agent C. Agent C will share $0.93 * 1.0 = 0.93$ with A. A has two options: $0.75 * 0.99 * 1.0 = 0.7425$ and $0.97 * 0.93 = 0.9021$. Since 0.9021 is greater, A shares that value with S_1. In making a decision S_1 will either pick A (knowing the probability of messages to get through to be 0.866) or pick B (assuming the probability of getting the message across is 1).

The disadvantage of this technique is it takes the next exploration to make a global decision and a local decision might not always be optimal. The second disadvantage is that it takes time to learn the probability of dropping messages on any link.

5. EXPERIMENTAL ANALYSIS

All experiments were conducted using NS2 [6], a standard network simulator. NS2 provides models for standard network link dynamics as well as message interference, both of which strongly govern available bandwidth. We also used the widely used NS2 implementation of OLSR developed at University of Murcia.[2] In our experimental analysis, we evaluate the benefit of adding organization knowledge to the lower level routing algorithm (eCQRouting). We measure: 1) the effect on performance with increasing number of exploration messages, 2) the scalability of the algorithm as the number of non-application agents increase, 3) the scalability of the algorithm as the density of agents increase, and 4) the effect of bandwidth available on messages with different priority.

5.1 The effect of exploration on performance

This experiment used the CNAS organization structure on the 1-D network shown in Figure 5.

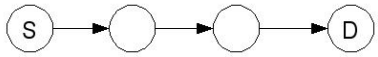


Figure 5: 1D network

Here S represents a sensor agent and D a cluster head agent. Next, neighbors that are not used by the high-level application and thus not part of the organization structure, are added to both the sensor agent and the cluster head in order to increase the number of exploration messages on the path from S to D. The neighbors are added such that they do not add additional paths from the source to the destination (Figure 6 shows an example network after adding 3 neighbors).

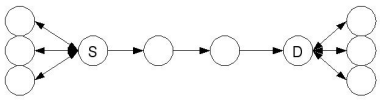


Figure 6: Adding 3 neighbors to both source and destination in Figure 5

With no additional neighbors, the maximum bandwidth available if no exploration were to be done was determined to be 180Kbits/sec. Agent S sends 180 1Kbit messages to agent D every second for 150 seconds. The average number of messages that reach agent D every second determines the bandwidth. We performed 10 runs and averaged the bandwidth over the 10 runs. The network is completely stable: no links fail during the course of the experiment. As additional neighbor agents are added to the network, traditional

[2]http://masimum.dif.um.es/?Software:UM-OLSR

routing algorithms (OLSR and DSDV[3] in our experiments) use additional exploration messages in order to include these agents in the routing tables of all other agents in the network. eCQRouting prevents this from happening by taking advantage of the CNAS organization knowledge that specifies that agent D to be the only destination agent in the network. The effect on the bandwidth available to agent S in sending its data to agent D is shown in Figure 7

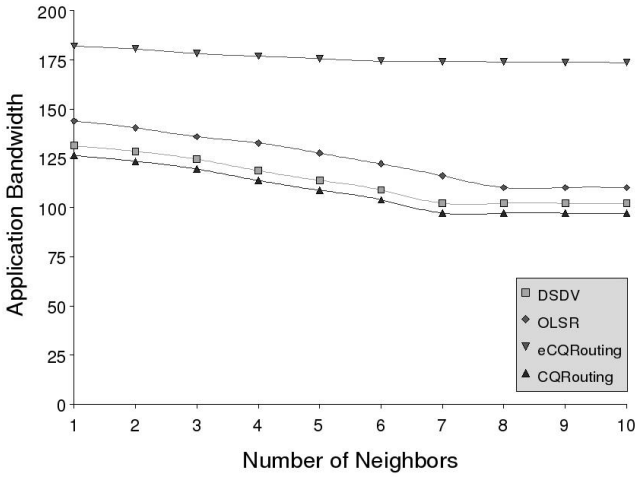


Figure 7: Changes in bandwidth as the number of neighbors (and in-effect the number of exploration messages) increases. Bandwidth is calculated in KBits/sec and averaged over 150 seconds

As the number of neighbors increases, the application level bandwidth available when using eCQRouting is significantly better than the next best algorithm (OLSR), with performance improvements of 20.9% with one neighbor to 36.55% with 10 neighbors. The algorithm also provides 30.5% additional bandwidth with one neighbor over CQRouting, which increases to 44% improvement with 10 neighbors. The reason for this improvement is the reduced number of exploration messages in eCQRouting. eCQRouting uses 15% of the number of exploration messages in OLSR with one neighbor, which drops to 7% with 10 neighbors.

5.2 Scalability in number of agents

We again use the CNAS organization, however the agents are now randomly placed within a predefined area. Networks of size 4 through 100 were evaluated. The density of the network remains the same throughout. Three sensor agents and one cluster are head placed randomly along with other agents as needed. The network is no longer stable: a link between two agents fails randomly every second with a probability of 0.2.

As the size of the network increases, the number of exploration messages also increases. However, since the density of the network remains the same, the number of hops to the destination also increases. The algorithm needs to be able to generate paths that are viable over the longer distances. Each sensor agent sends 180 1Kbit messages to the cluster head agent every second for 150 seconds. We measure the average number of messages received by the cluster head every second; which defines the application-level bandwidth for the run. Figure 8 shows the effect of this increasing size on the bandwidth averaged over 10 runs.

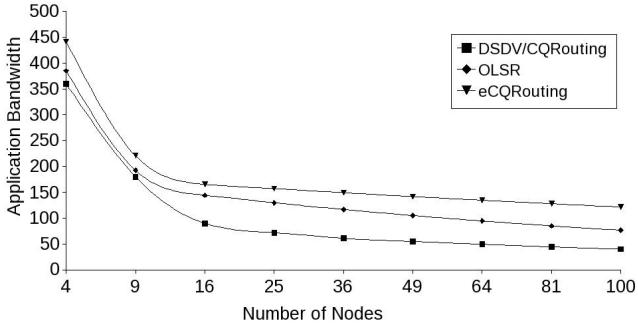


Figure 8: Changes in bandwidth as size of the grid network increases. Bandwidth is the average bandwidth over 150 seconds, measured in KBits/sec

Figure 8 shows an initial drop in the bandwidth available, as the destination agent is no longer 1 hop away from the sensor agents. The reduction in application-level bandwidth diminishes as additional hops are added between the source and the destination. The performance improvement in this experimental setting of eCQRouting over OLSR range from 10% for the 4 agent network to 35% for the 100 agent network. The improvement over CQRouting goes from 16.5% for the 4 agent network to 66% for the 100 agent network. Both DSDV and CQRouting have similar performance due to similar source-based exploration algorithms. As the number of hops increase, the time to converge to the optimal path also increases.

5.3 Scalability in agent density

We start with the 25 agent CNAS layout defined in the previous section. We keep the area constant, and add 25 to 200 agents at random positions within the network. This increases the density of the network, which in turn increases the number of paths from the source agents to the destination agent. Figure 9 shows the average application bandwidth of 20 runs for each density variation.

Figure 9 shows eCQRouting performs about 17% better than OLSR in the 25 agent network with minimal density. When density is doubled, the performance improvement increases to 19%. At triple the density, the performance improvement is 10%. When the density reaches 8 times the minimal density, OLSR outperforms eCQRouting. This is because the benefit of building MPR agents outweigh its costs at this density, allowing OLSR to do much better than all other routing algorithms.

5.4 Effect of the bandwidth available on messages with different priority

To explore the effect of priority, we used the CASA organization with priorities on messages as shown in Figure 2b. In particular, we look at networks where the MAS can tolerate the loss of a certain percentage of messages with-

[3]DSDV is a table driven routing algorithm for adhoc wireless networks based on the Bellman Ford Algorithm. Its similar to CQRouting in that agents share routing tables (rather than sharing QValues) with their neighbors and use the shared tables in developing local routing algorithms.

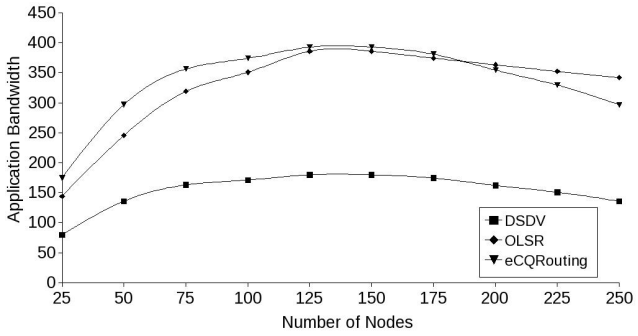


Figure 9: Changes in bandwidth as network density increases. Bandwidth is the average bandwidth over 150 seconds, measured in KBits/sec

out significant loss in performance. We use curves shown in Figure 10a. The performance degradation with message lost from the Feature Detector (FD) agent to the Feature Repository (FR) and messages lost from FR agents to Optimizer (Opt) agent is linear. Messages from Radars (Rad) to FR agents have some slack, which allows for a greater percentage of messages to be dropped before there is a significant effect on performance. Messages from Optimizers to Radars have almost no slack, and there is a significant drop in performance with drop in messages.

Since messages from FD to FR and Rad to FD have the same performance degradation, we gain no additional benefit by analyzing the two separately, and can perform better analysis by merging the role of FR and FD into a single role (calling it the FR/FD role). The priority is averaged and the modification is shown in Figure 10b

We use a sparse 16 agent network. The network is divided into 4 clusters. The density of the network is increased by keeping the area constant, but moving from 16 agents to 160 agents, placed randomly. An example 25 agent network is shown in Figure 11 (agents on the border between two clusters belong to the cluster where their center lies). For each cluster, we randomly assign the role of Opt to one agent and the role of FR/FD to another agent. The rest of the agents (unmarked in the figure) are all Radar agents. The Rad agent sends 100 1Kbit messages to the FD/FR agent every second. FR/FD agents send 25 1Kbit messages/second to the Opt agent and Opt agents send 5 1Kbit messages/second to the Rad agents.

In this experiment, we start with enough bandwidth to allow all messages to get through. Next, the bandwidth is reduced until a certain percentage of messages are dropped for each routing algorithm. For example, the threshold before 10% of messages are dropped is different for different routing algorithms. The effect of dropping messages, is calculated at different bandwidths. Figure 12 shows the performance degradation with messages loss. Note, the global performance across the three message types is calculated as the product[4] of individual performances.

Figure 12 shows that the global performance is much higher for the eCQRouting algorithm. This is because the low priority messages are dropped before high priority messages. OLSR, DSDV and CQRouting all performed the same when dropping the same percentage of messages, since the message loss is equivalent across all messages types for each of the three algorithms. Interestingly, no improvement was obtained using either one of the two priority techniques described in Section 4.2 (represented as eCQRouting with priority for the simple priority based measure and eCQRouting with performance measure for the message-loss based approach). This is primarily because both algorithms attempt to do the same optimization on the network. The differences arise in when the evaluation occurs and the degree of evaluation. With the priority based representation, the status of network and the goodness of the path are evaluated at the time of exploration. Exploration happens more frequently for source-destination pairs where the priority of messages is much higher, hence allowing them to find sub-optimal paths that provide better probabilities of message delivery. However, if the state of the network changes between exploration messages, the technique will have a delayed reaction to the change. On the other hand, the per-message decision is based on local knowledge of the next hop and expectation of the rest of the network. However the benefit of having global information is lost. In this experiment, it appears

[4]For simplicity, the performance at each agent is assumed to be independent of other agents, and hence multiplied to give us global performance

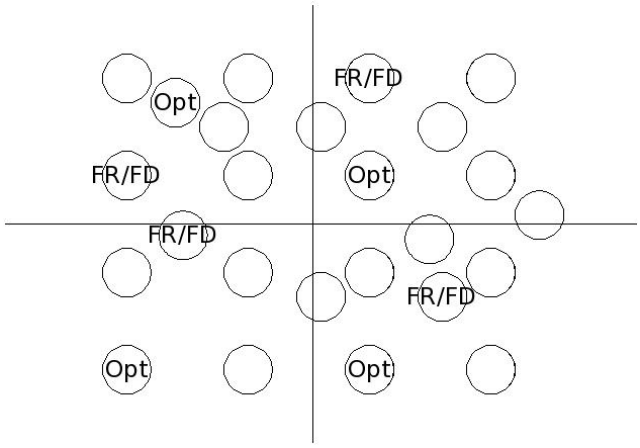


Figure 11: Example 25 agent network

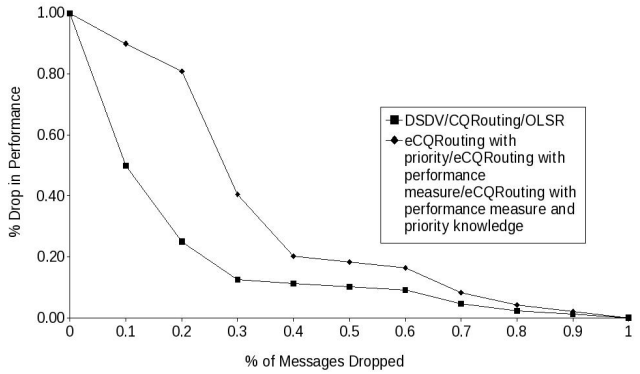
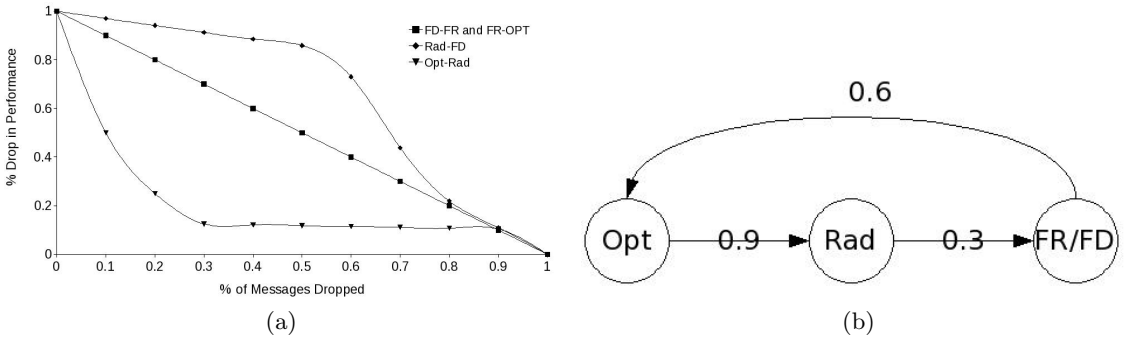


Figure 12: Performance degradation with loss of messages



(a) (b)

Figure 10: (a) Performance degradation as messages get dropped for messages from FD agents to FR agents, FR agents to Optimizer agents, and Optimizers agents to Radar agents. (b) Corresponding priority of messages

that the global and local benefit balance each other, resulting in similar results when used individually and a small 2% improvement when combined.

6. CONCLUSION AND FUTURE WORK

In this paper, we showed that using organization-level knowledge in network-level routing significantly improves application-level bandwidth relative to standard routing protocols. In a 100 agent simulation of the CNAS sensor network, we obtained an increase in available application bandwidth of 35% as compared to OLSR, one of the more widely used routing algorithm for wireless ad hoc networks. Additionally we observed an increase in global utility by focusing path exploration and, in turn, generating better paths for those source-destination pairs where communication has higher priority.

In this work, information flowed from the application-level organization to the lower-level routing algorithms. We assumed that the organizational structure was appropriate given the network structure. If we relax this assumption, an obvious extension of this work would be to allow the organizational structure to be modified based on the changing characteristics of the network. The network-level routing algorithm produces network capability and status information that can be used as input for possible adaptation or redesign of the organizational structure. Furthermore, as the organization structure changes, the updated information would be passed to the routing algorithm, increasing application bandwidth further. For example, cluster heads are picked dynamically in CNAS. However, we made the assumption that the cluster head was already chosen in the organizational knowledge provided to the routing algorithm. In practice, the routing layer could provide the sensor agent with a list of all other agents that it can currently communicate with that are within its cluster area. A decision of which of these agents is the cluster head could be made and then provided to the routing algorithm. Moreover, with the increasing research in emergent organizational design, a MAS application can start with no organizational structure, use the routing information to develop an initial organization design, which then provides exploration direction to the routing algorithm for better paths and bandwidth.

7. REFERENCES

[1] J. Boyan and M. Littman. Packet Routing in dynamically changing networks: A reinforcement learning approach. *Cowan, J.D.;Tesauro, G.;and Alspector, J., eds., Advances in Neural Information Processing Systems*, 1994.

[2] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol. *RFC 3626, Internet Engineering Task Force (IETF)*, October 2003.

[3] D. Corkill, D. Holzhauer, and W. Koziarz. Turn Off Your Radios! Environmental Monitoring Using Power-Constrained Sensor Agents. *First International Workshop on Agent Technology for Sensor Networks (ATSN-07)*, 2007.

[4] D. Corkill and V. Lesser. The use of meta-level control for coordination in a distributed problem solving network. *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 748–756, August 1983.

[5] B. Horling and V. Lesser. A Survey of Multi-Agent Organizational Paradigms. *The Knowledge Engineering Review*, 19(4):281–316, 2005.

[6] S. Keshav. REAL: A Network Simulator. *tech. report 88/472, University of California, Berkeley*, 1998.

[7] M. Krainin, B. An, and V. Lesser. An Application of Automated Negotiation to Distributed Task Allocation. In *2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2007)*, Fremont, California, November 2007.

[8] S. Kumar. Confidence based Dual Reinforcement Q-Routing: An On-line Adaptive Network Routing Algorithm. *Master's thesis, Department of Computer Sciences, The University of Texas at Austin, TX-78712, USA Tech. Report*, A:198–267, 1998.

[9] J. McQuillan and D. Walden. The ARPANET Design Decisions. *Computer Networks*, 1, August 1992.

[10] R. Onishi, S. Yamaguchi, H. Morino, H. Aida, and T. Saito. A multi-agent system for dynamic network routing. *IEICE Transactions of Communications*, 84-B(10):2721–2728, 2001.

[11] C. Perkins and P. Bhagwat. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. *ACM SIGCOMM Computer Communication Review*, 24(4):234–244, October 1994.

[12] C. Perkins and E. Royer. Ad-hoc On-Demand Distance Vector Routing. *Proc. 2nd IEEE Workshop. Mobile Comp. Sys. and Apps*, pages 90–100, 1999.

[13] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1989.