1-6-2009

Agent Sensing In Limited Resource Environments

Adam Eck
University of Nebraska, aeck@cse.unl.edu

Leen-Kiat Soh
University of Nebraska, lsoh2@unl.edu

University of Nebraska–Lincoln, Computer Science and Engineering
Technical Report TR-UNL-CSE-2009-0005
Issued Jan. 6, 2009

Agent Sensing In Limited Resource Environments

Adam Eck Leen-Kiat Soh
Department of Computer Science and Engineering
University of Nebraska-Lincoln
256 Avery Hall
Lincoln, NE 68588, USA
+1-402-472-4257
{aeck, lksoh}@cse.unl.edu

ABSTRACT

One of the key challenges for multiagent systems (MAS) is optimizing performance in limited resource environments. Previous research in this area has focused on the problems of 1) resource allocation and arbitration, and 2) bounded rationality, which describe the relationship between resource constraints and both agent reasoning and actuation. However, less work exists addressing the effect of consuming resources during agent sensing, particularly two important tradeoffs. First, sensing can reduce resource availability, resulting in a tradeoff between overall system performance and an agent's sensing behavior (the *Performance Tradeoff*). Second, consuming resources during sensing can alter the outcome of the measurement (the *Observer Effect*). Since an agent requires up-to-date information, but tracking too frequently can worsen the observer effect, there also exists a tradeoff between the quality and frequency of an agent's sensing (the *Information Quality Tradeoff*). We present an algorithm for Resource-Aware Tradeoff-based Sensing (RATS) which considers trends in both the need for information and system performance to *learn* an appropriate sensing frequency. The agent considers a sliding window of possible frequencies bounded to avoid decreases in system performance while providing quality information and chooses an appropriate frequency based on its confidence in sensing. To validate our algorithm, we conducted experiments with 30 agents in a simulation of agent-based wireless networks, with different levels of resource constraints, to compare RATS sensing against only-need-aware sensing and only-performance-aware sensing. Our results show that RATS agents experience better system performance than only-need-aware sensing, while producing more accurate models than only-performance-aware sensing.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *intelligent agents, multiagent systems.*

General Terms

Algorithms, Experimentation

Keywords

Agent Sensing, Resource-Awareness, Observer Effect

1. INTRODUCTION

One of the fundamental challenges for multiagent systems (MAS) is optimizing performance in lieu of resource limitations imposed by various hardware, software, and human constraints. These constrained resources include computational cycles, memory, network bandwidth, time, knowledge, user skills, etc., resulting in problems such as sub-optimality of both task solutions and agent reasoning, contention for scarce resources, and even deadlock.

Previous research on resource limitations within MAS has taken two primary directions: 1) *allocation* and *arbitration* of resources, and 2) *bounded rationality*. The former includes work on distributing scarce resources efficiently and optimally to agents from a global perspective using both centralized and distributed approaches [7]. Allocation also involves assigning resources from a local perspective amongst an agent's tasks to optimize the utility of the agent's actions [1]. Research on bounded rationality, on the other hand, studies how to efficiently control agent reasoning under bounded computational resources [12, 18].

While such research explores the implications of resource limitations on both agent reasoning and actuation, less present in the agent literature is work addressing the relationship between resource consumption and sensing. In order for agents to gather information about limited resources, they must often consume resources, *including the resource being tracked*. This additional expenditure of limited resources results in two primary problems: 1) it reduces resource availability both for other agent activities from the local perspective and for allocation to all agents from the global perspective, *affecting system performance*, and 2) consuming a resource during sensing alters the agent's measurement, *affecting the result of each local observation*. This latter problem is known in the physical sciences as the **observer effect**, which states that the simple act of making an observation alters the outcome of the observation. For example, tracking the computational resources consumed by an agent requires additional CPU cycles, inflating the measured value. Similarly, sending messages between agents to track network conditions increases traffic, again altering measured conditions. Thus, sensing in resource bounded environments entails two key tradeoffs: 1) between overall system performance and the agents' sensing activities since sensing consumes valuable limited resources, and 2) between the quality of observations and the rate and quantity of sensing because the observer effect results in a different environment state than would have otherwise occurred.

These tradeoffs are especially problematic in MAS because they increase the **uncertainty** of an agent's beliefs about both whether or not its beliefs reflect the dynamic environment, as well as the impact of the agent's sensing on the environment. As an agent increases its sensing, it continually collects more up-to-date information to base its beliefs upon, generally increasing certainty that these observations reflect the current state of the environment. However, due to the dynamic and teleological behavior of the environment, an agent cannot deterministically calculate how an increase in sensing will influence its environment or its mea-

surements through the observer effect. Conversely, if an agent decreases its sensing, the certainty that its beliefs are up-to-date also decreases, while the agent becomes more certain that its sensing is not affecting the environment or altering its observations. This situation is analogous to **Heisenberg's Uncertainty Principle** [11] where increasing the certainty of one belief decreases the certainty of another.

To balance the tradeoffs resulting from sensing in limited resource environments, we have developed an algorithm for Resource-Aware Tradeoff-based Sensing (RATS) to automatically adjust the amount of sensing performed by agents depending on the perceived environment state, the agents' confidence in their gathered data, and the need for up-to-date information, in order to reduce the impact of their sensing on the environment. This algorithm focuses on controlling sensing from each agent's local perspective, relying on local decisions to generate the coherent emergent behavior of system-wide resource-awareness and management while minimizing communications between agents, accounting for the fact that communication resources could also be limited. The algorithm is computationally inexpensive, allowing it to be used with bounded rational agents (e.g., robots, sensors). To validate our algorithm, we have conducted comprehensive experiments with 30 agents to balance sensing in a simulated wireless network supporting an online collaboration environment where agents model shared wireless resources and adapt their sensing to avoid contention with collaborating users. We have observed that by considering both tradeoffs imposed by multiagent sensing in limited resource environments simultaneously, RATS agents gather more accurate information than only considering either of the tradeoffs, and experience improved system performance by considering the tradeoff between system performance and sensing.

This research fits the current study of multiagent systems in the following ways: 1) its focus lies along the intersection between agent perception, resource-aware reasoning, and the effect of agent-environment interactions; 2) it describes experiments conducted to evaluate a new algorithm to solve the problems caused by multiagent sensing in limited resource environments, and 3) the problem was inspired by the observer effect and Heisenberg's Uncertainty Principle from the physical sciences domain and our solution is inspired by localized agent learning leading to coherent, emergent, multiagent behavior.

The rest of this paper is organized as follows. Section 2 provides some necessary background and related work. In Section 3, we present our RATS algorithm to balance the tradeoffs of multiagent sensing in limited resource environments. Next, in Section 4, we describe the experiments conducted to validate our algorithm, including an overview of our application and simulation testbed, followed by the accompanying results in Section 5. In Section 6, we provide a discussion of the results, focusing on the lessons learned through our experiments and the implications for resource-aware sensing in real-world environments. We conclude with a brief summary and important future work in Section 7.

2. BACKGROUND AND RELATED WORK

Across a diverse set of fields, previous research related to resource-aware, multiagent sensing includes work on 1) *sensing for resource-aware applications*, 2) *considering the costs associated with sensing*, and 3) *the control of agent sensing*. First, the application of resource-awareness to the networking domain is relevant to this paper because it defined the notion of active and passive monitoring. Active monitoring occurs when a monitoring entity (e.g., an agent) injects additional packets into the network to gather information, while passive monitoring occurs when the entity extracts information from already existing traffic [13]. The data gathered through both types of monitoring is then used to adapt the application to the current state of the network (e.g., changing the encoding of multimedia content [5]). Active and passive monitoring has also been studied with mobile agents [6], and similar resource-aware multimedia systems include the MAS Raja [8].

The notion of active/passive monitoring can be extended from the networking domain to multiagent systems in the form of *active and passive sensing*, where agents either perform specific sensing tasks to gather information (often consuming the resource being tracked), or extract information from system activities, respectively. Existing *hybrid* monitoring systems [13] aim to rely on passive monitoring to reduce the burden on the system imposed by monitoring, but use active monitoring when necessary. These systems rely on static rules that define when active monitoring should be employed based on the application or network (e.g., when passive information is unavailable [13]). We take a similar approach, but generalize to any domain or sensing activity and use *learning* to adapt the agent's sensing behavior to reduce strain on the environment while maintaining accurate information. We also consider the *observer effect* which states that the accuracy of active sensing is reduced through consuming limited resources.

Second, previous work has considered the costs of gathering information. Within MAS, researchers have studied combining multiple agents' sensed data to avoid costly interruptions of users in fast-paced environments [19], where information must be quickly processed but is used infrequently. Our work also attempts to avoid costly data acquisition but produces information that is used frequently throughout the operation of the system. We also consider the potentially high costs of agent communications in limited resource environment. Other research reduces the costs associated with sensing in wireless sensor networks, especially energy costs. In [17], a similar approach to our methodology is taken where agent-based sensors adapt their sensing frequency using a window of possible frequencies. Our work is very similar, but differs in several key ways. First, our research focuses on the effect of sensing on the *shared* resources being monitored, not just the *local* resources needed for sensing. Second, the bounds of our window are adapted to the environment, while theirs are statically set by users. Third, in [17], the agent automatically switches to the maximum sensing frequency whenever a change in the environment is detected, while we consider that such drastic increases in sensing can produce subpar results and hurt overall system performance. Finally, their work is more appropriate for wireless sensor networks because it interleaves sensing costs and routing in multihop networks. In the MAS-related field of autonomic computing, recent work has begun recognizing the impact of processor load monitoring on the load of the processor, specifically that the cycles must be consumed to monitor the resource, changing the observation (i.e., the observer effect) [3].

Lastly, research has also been conducted on controlling agent sensing. Specifically, the use of anytime algorithms has been applied to agent and robotic sensing to gather *enough* information [10, 24]. Our research also aims to control sensing to gather quality information, but we recognize that, due to the observer effect,

the quality of sensing is not monotonically increasing with respect to the amount of sensing.

3. METHODOLOGY

3.1 Environment Characteristics

The environment considered for our methodology is one that consists of multiple autonomous agents performing actions in real-time to meet a set of (possibly conflicting) goals. Achieving these goals requires consuming limited resources, leading to contention for resources between agents. Agents can communicate with their peers; however, resources required for communications can be limited, and the costs of communication vary with the contention for their required resource.

In order to make appropriate real-time decisions for working with limited resources, agents gather information about their environment, which is then used to construct models which represent the current state of the environment and guide an agent's reasoning process. Agents can gather information using two techniques: *active sensing* and *passive sensing*. Each of the two information gathering activities has its advantages and disadvantages. Active sensing is beneficial because it allows an agent to directly measure a feature of the environment, and this information can be acquired on-demand, whenever necessary. However, when measuring a limited resource with active sensing, **the agent must consume an additional amount of this resource to acquire the information it desires, potentially changing the behavior or availability of the resource.** Passive sensing, on the other hand, is valuable because it does not require any additional consumption of resources to gather information about a resource, and for active systems, it can provide a large quantity of information. The downside to passive sensing is that it only occurs with other tasks, so if the environment is dormant or activities occur infrequently, up-to-date passive monitoring information is unavailable. Also, passive monitoring might not directly gather the information required by an agent, instead providing a rough approximation.

These qualities of the environment can be summarized with the following characteristics: (1) the environment is dynamic, real-time, and teleological; (2) resources required for completing tasks suffer from limited availability, leading to contention; (3) using resources for sensing affects the status of the resources, and subsequently, the environment; (4) active sensing provides more accurate data because it directly measures the values necessary for the agents' reasoning; (5) passive sensing provides intermittent information which approximates the values observed through active monitoring; and (6) communication costs vary with resource limitations and can be large in times of high contention.

These characteristics lead to several problems for agents within the environment. First, the agents' information gathering activities can *affect resource availability in the system*, leading to a *tradeoff between the quality of system performance and the amount of sensing performed by the agent* (the Performance Tradeoff). Second, the consumption of limited resources during active sensing leads to the observer effect, where the consumption of the tracked resource during the measurement changes its outcome. *We assume that the greater the contention, the more impact each additional consumption has on the resource, leading to a greater discrepancy due to the observer effect.* Combined with the fact that agents require up-to-date information for decision making in real-time systems, this leads to a *tradeoff between the quality of observations and the frequency of the agent's sensing* (the Information Quality Tradeoff) since more frequent sensing leads to greater contention, while less frequent sensing can result in stale information.

Together, these tradeoffs entail two levels of uncertainty in an agent's beliefs. First, the more frequently an agent senses its environment, the more certain it is that its beliefs about the environment's state are up-to-date. This increase in sensing also increases contention for limited resources and could decrease system performance and increase the observer effect. Therefore, an increase in tracking can decrease an agent's certainty that its monitoring is not hurting the environment or the quality of its measurements. Likewise, decreasing tracking decreases an agent's certainty that its beliefs reflect the current state of the environment, while the agent can be more certain that its lower quantity of sensing is not adversely affecting the environment since this behavior results in less contention for limited resources. **Thus, balancing these tradeoffs is a key problem for sensing in limited resource environments.**

3.2 RATS Algorithm

To solve the problem of balancing agent sensing against both system performance and the quality of information gathered during sensing, we have developed a methodology called Resource-Aware Tradeoff-based Sensing (RATS) that simultaneously considers the affect of sensing on both tradeoffs. In RATS, each agent performs active sensing at an adjustable interval (i.e. periods). The set of possible intervals corresponds to a *sliding window*, adjusted over time to account for the current state of the environment. This window is bounded on one end by a limit to avoid decreased system performance (the Minimum Interval Bound – *MIN_INT*), and on the other end by a limit to provide up-to-date information (the Maximum Interval Bound – *MAX_INT*), as shown in Figure 1. As both the need for up-to-date information and system performance vary over time, the agent adjusts these bounds to *adapt* to changes in the dynamic environment. Within the window, the agent *learns* an appropriate interval based on its *confidence* in the information gathered during both active and passive sensing. The process for performing RATS is described in the following subsections, and is given by Algorithm 1: *The RATS Algorithm*. Please note that we use intervals in this algorithm instead of frequency (which are equivalent between the time and frequency domains) to simplify scheduling.

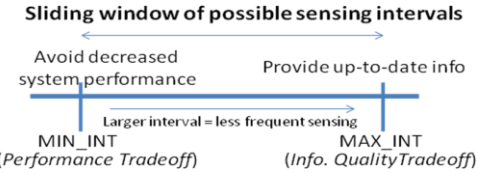

Figure 1: RATS Sliding Window of Sensing Intervals

Note that as shown in Figure 1, the novelty of the RATS algorithm for resource-aware sensing lies with how the agent balances the two tradeoffs. Each tradeoff relies on the other to provide a "check" on its influence. The Performance Tradeoff determines how much to sense so as to avoid negatively impacting the system performance. Left unchecked, however, an agent would attempt to sense as little as possible. Thus, an information quality need is used to motivate more sensing. On the other hand, the Information Quality Tradeoff determines how much to sense so as to have

up-to-date information about the system. Once again, if left unchecked, an agent would attempt to sense as much as possible. Thus, we use the observer effect principle (via the deterioration of system performance) to motivate less sensing.

3.2.1 Performance Tradeoff

To account for the tradeoff between system performance and agent sensing, the agent maintains a Minimum Interval Bound on its active sensing interval to avoid consuming too many resources during sensing. If sensing were to near or exceed this bound, the agent would expect to increase the contention for limited resources by sensing too often, reducing availability for other tasks and agents, thereby hurting overall system performance. From another perspective, this bound also limits an agent's uncertainty in the effect of its sensing on the environment – as the agent stays above this bound, it knows that its affect is minimal, but crossing the bound could hurt the environment by some unknown amount.

To adapt to changes in the dynamic environment, the agent periodically adjusts *MIN_INT* on the sliding window (Step 1 of RATS). First, the agent must approximate the current level of system performance using its model of the environment (Step 1.1). The metric (e.g., network latency, processor throughput, etc.) and calculation for this step depend on the application employing RATS. Next, the agent compares the current performance value (*new*) to the previous value (*old*) to determine a normalized change percentage using Eq. (2) (Step 1.2).

$$change = max(-1.0, min(1.0, (new - old) / new)) \quad (1)$$

Then, the agent can compute the amount to shift *MIN_INT* (Step 1.3) using an *aggressiveFactor* to represent how fast the bound can shift in either direction as in Eq. (3).

$$shift = change * aggressiveFactor \quad (2)$$

3.2.2 Information Quality Tradeoff

To account for the tradeoff between sensing quality and frequency, the agent also maintains a Maximum Interval Bound on its active sensing interval to avoid stale, out-of-date information. Similar to the system performance bound discussed previously, if the sensing interval were to move beyond *MAX_INT*, the agent would be sensing too infrequently and should expect its information to be out-of-date and of low quality, while sensing more often (i.e. at a smaller interval) should produce information which accurately reflects the current state of the environment, increasing its certainty in the quality of its beliefs. To adapt sensing to provide data accurately capturing environment state, the agent begins by approximating the current need for up-to-date information for modeling the environment and reasoning about tasks. This need results from two qualities of sensing: 1) stability of models, and 2) situation specific information. First, if the models produced by the agent are stable over time, the agent can assume that the environment is relatively static, so older information is not becoming stale, reducing the need for more frequent tracking. Second, if an agent falls into (application-specific) special situations, it could need to perform additional active sensing compared to its normal operation to make important decisions, increasing the need for active sensing. To compute the stability of models, RATS uses the Wilcoxon Rank Sum Test [22] which compares two data sets to determine if they are governed by the same probability distribution. Considering sensing observations and model values as random variables, if the current model's values come from the same distribution as the previous models (with quantity *windowSize)*, the previous models capture the same information as the current model, so the environment is relatively static. The Wilcoxon Test was chosen because it makes no assumptions about the underlying distributions for the data sets, allowing the test to be applied to nearly any environment, and it is computationally inexpensive to compute [19]. The agent's confidence in stability is calculated in Eq. (3), where the current model's values are α, and the previous models' values are β, and the *p*-value of the test is the confidence that the two data sets do *not* come from the same distribution.

$$Confidence = 1 - pValue(WilcoxonTest(\alpha, \beta)) \quad (3)$$

The calculations for need arising out of situation-specific information, on the other hand, is application specific. To combine these components, the agent takes a weighted average between each of the stability and situation-specific need values as its total need.

As with *MIN_INT* due to the Performance Tradeoff, the agent then compares the current and previous values for need to create a normalized change percentage using Eq. (1) and shifts the bound using the same *aggressiveFactor* with Eq. (3) (Steps 1.2-1.3).

3.2.3 Balancing the Tradeoffs with Interval Selection

Once both bounds are computed, the agent is ready to select an appropriate sensing interval. First, it saves the current need and system performance for future consideration (Step 1.4) and compares the bounds to make sure they did not cross one another (Step 1.5). If they did, both bounds are set to the lowest of the two to avoid both bad system performance and the observer effect as much as possible. In future iterations of the algorithm, the two bounds are then free to move apart.

Then, the agent selects a new sensing interval within the sliding window (Step 2 of RATS). The new interval is based on the explicit *confidence* an agent has in the information gathered by its sensing activities, especially its passive sensing. As this confidence value increases, the agent needs less active sensing to accurately model the environment, relying instead of free passive sensing, and vice-versa. This consideration is critical to accommodating for the observer effect in RATS. If the models obtained via active and passive sensing are similar, then the observer effect is likely to be minimal. Therefore, once again, we use the Wilcoxon Rank Sum Test to compare the current models obtained by passive and active sensing, per Eq. (3) and generate a confidence value accordingly. With this new confidence value, the agent again computes a normalized change percentage from the confidence in the previous iteration using Eq. (2) (Step 2.2). If the

Algorithm 1: RATS Algorithm

Begin

1. Adjust tracking window bounds
   - 1.1. Compute current need and system performance
   - 1.2. Normalize change in values using Equation (1)
   - 1.3. Compute shift in bounds using (2), then move
   - 1.4. Save the current need and system performance
   - 1.5. If the bounds cross, set both equal to the maximum
2. Adjust tracking period within bounds
   - 2.1. Compute current confidence in data using Equation (3)
   - 2.2. Normalize change in confidence using Equation (1)
   - 2.3. Compute new interval using Equation (4)
   - 2.4. Save the current confidence value
   - 2.5. If the interval exceeds a bound, set it to the bound

End

change in confidence is positive, the agent is more confident so less sensing is necessary and the agent will select a larger sensing interval towards *MAX_INT*, or vice versa (Step 2.3). This new interval is computed using Eq. (4).

$$interval = interval + (selected\_bound - interval) / 2 * change \quad (4)$$

In this step, the agent only moves the interval up to the midpoint with the selected bound to produce more conservative changes as the interval nears a bound to avoid degrading performance and to not shift the balance of either tradeoff too greatly, since the effect of the tradeoffs are highest near the bounds.

Finally, the agent saves the current confidence value for future consideration (Step 2.4), and performs one last check to make sure that the interval is not beyond a bound (Step 2.5), which could occur if the bound originally shifted beyond the interval in Step 1.

4. EXPERIMENTS

To validate our methodology, we implemented the RATS algorithm to adapt sensing for agent-based wireless network monitoring in a simulated online collaboration environment (OCE). In OCEs, agents provide services such as matchmaking and user modeling/assessment (e.g., [4, 20]). To make resource-aware decisions, agents must gather information about limited wireless resources. Such an environment is well-suited for RATS because wireless network performance varies over time, producing differing level of contention for the network resource shared between users and agents. Active sensing of the resource occurs through sending special messages between agents to evaluate the status of the network, increasing congestion and latency in the network, which decrease user productivity. Agents can also use passive sensing to extract information (e.g., single trip latencies) from messages transmitted between users. Finally, because network resources are limited, communication costs between agents vary and can be quite large if contention is high.

To simulate a wireless network OCE, we built a multiagent simulator using the Repast Agent Simulation Toolkit [16] and the JavaStatSoft [21] software for the Wilcoxon Rank Sum Test. The wireless network is modeled as a two-state Markov process, switching between periods of loss and successful transmission to simulate bursty loss behavior using the parameters for a standard network from [15]. Contention for the network is created by limiting the number of messages which can be transmitted during a given period of time (20 per 20 ms simulation tick), simplifying the CSMA/CA-based MAC protocols used for wireless networks [23]. We use the PGM protocol [14] to control the transportation of messages, which provides reliable multicast delivery in lossy wireless networks [9]. We simulate OCE traffic by generating dialog-based messages within groups (of 5 users), where the delay between sending responses from a single user follows a Gaussian distribution with a mean/median of 15 seconds, matching the median observed in [2].

In our simulations, system performance is measured in terms of the *latency* of the network, calculated as the amount of time a message and its response spend in the network before reception by both parties. This measures the overhead in the network for collaboration between users and varies with loss and congestion, and is estimated by agents during active sensing. The single trip latencies of each message observed in passive sensing also (less accurately) approximate this value. The need for quality information depends on the stability of the agent's models (*windowSize* = 5, created every minute of simulation time), as well as the special situation for user modeling where an agent must determine if responses from users are unsent or unreceived due to high latency.

To evaluate RATS sensing, we conducted experiments with 30 agents and users to compare four agent sensing behaviors: no sensing, only-need-aware (NA), only-performance-aware (PA), and RATS. NA agents only consider the Maximum Interval Bound on sensing intervals and ignore system performance, whereas PA agents only consider the Minimum Interval Bound due to performance while ignoring need. The *aggressiveFactor* for shifting the sliding window bounds was set to 10 seconds. Adjustments in the sliding window of sensing intervals were considered right before every new model was generated. We varied the amount of background traffic to create different levels of resource contention (starting where the worst behavior began to experience contention, *up to* a level where every behavior suffered) and ran the experiments for a half hour of simulated time. However, if the network became too congested to support collaborative traffic (indicated by a threshold of only 15 total user messages received in a minute of simulation time), the simulations were also stopped. Finally, each experiment was run 30 times using different random seeds to reduce variance in the results. We collected information about the latency in the network to evaluate the Performance Tradeoff, along with the accuracy of the agents' models against the true network state to evaluate the Information Quality Tradeoff. We recorded the duration of each experiment to evaluate the effect of the sensing behaviors on network congestion.

5. RESULTS

5.1 Performance Results

To analyze how well the RATS algorithm balances the Performance Tradeoff, we present the average latency for message-response pairs in the network as a function of external traffic percentage (indicating the level of contention for the resource) and the type of sensing performed by the agents in Figure 2. We also present the average duration of each experiment in Figure 3. Our simulations ran for at most one half hour of simulation time, which for a ratio of 20 ms per tick, results in a maximum duration of 90,000 ticks. However, these simulations also ended early when simulated users experienced unacceptable levels of network congestion due to wireless resource contention, so lower average durations also identify worse system performance.

From these figures, we can make several important observations. First, as contention for wireless network resources increased, system performance decreased for all agent types, as indicated by an increase in latency in Figure 2 and lower average durations in Figure 3. Furthermore, the decrease in performance for sensing agents occurred at a greater rate as contention increased than for the baseline no tracking experiments. **Thus, as contention increases, sensing has a greater impact on overall system performance, confirming the Performance Tradeoff.**

Second, only-need-aware (NA) agents suffered worse latencies than only-performance-aware (PA) agents and RATS agents. This is due to the fact that NA agents do not consider the impact of sensing on the environment, so the increases in sensing caused by the need to gather quality information is not checked by *MIN_INT*. In contrast, PA and RATS agents achieved lower latencies by considering *MIN_INT***,** creating larger intervals between

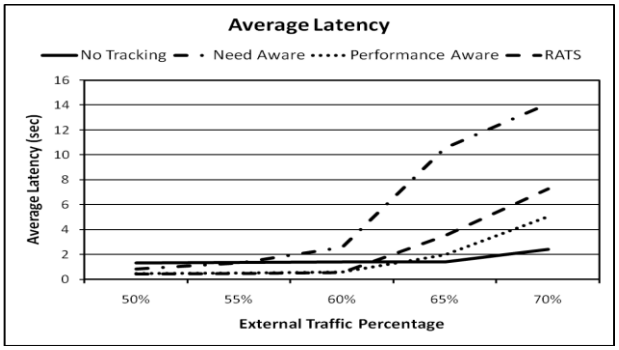

Figure 2: Average Latency of Message Traffic

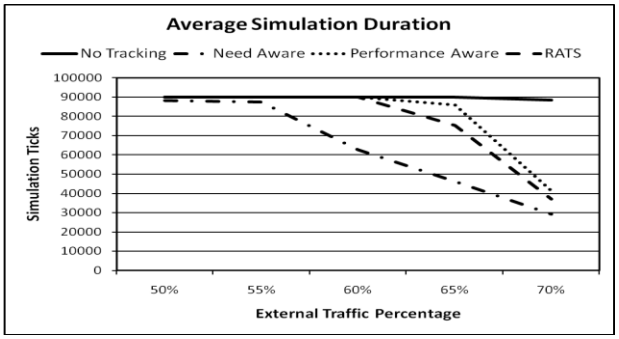

Figure 3: Average Simulation Duration

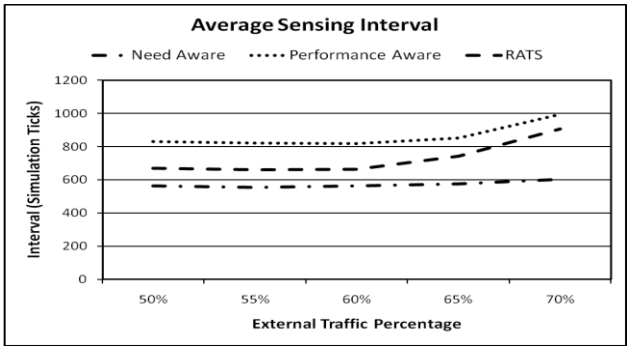

Figure 4: Average Sensing Intervals

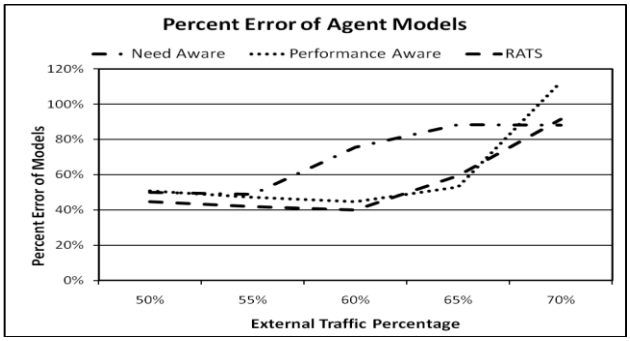

Figure 5: Percent Error of Agent-based Network Models

sensing activities. This discrepancy is evident in Figure 4 which shows that NA agents had lower average sensing intervals during all experiments, while PA and RATS agents naturally waited longer between sensing and further increased their intervals during worsening contention (i.e., 65% an 70% external traffic) to avoid hurting the environment. For NA agents, we can also observe that the simulations lasted a shorter period of time than the other agents, dropping off sharply at only 60% external traffic, indicating high levels of additional contention caused by unbounded sensing, while PA and RATS sensing experienced more external contention before unacceptable congestion.

Third, in terms of both network latency and simulation duration, PA agents outperformed RATS agents because RATS agents sense with smaller intervals (i.e., more frequently) and thus cause more contention for resources than PA agents. However, the performance of RATS agents was much closer to PA agents than NA agents, providing evidence that considering the Information Quality Tradeoff simultaneously at least in some environments does not have a large impact on the Performance Tradeoff.

Finally, we also observe the curious result that for the lowest levels of network contention, all three sensing behaviors actually resulted in lower latencies (i.e., *better* system performance) than the baseline no sensing agents. This seems at odds with the Performance Tradeoff because an increase in sensing over no sensing caused *improved* system performance. However, investigating further, this result is due to the behavior of the PGM network protocol used to transport user messages. In wireless networks, one primary cause of latency is the time needed to detect and re cover lost packets through retransmissions [14]. Due to its sequential ordering of packets, PGM detects loss only when a later packet is received. Agents using active sensing generate more packets than agents with no active sensing, and this increase in packets results in a faster discovery of loss, leading to a faster recovery and lower latencies from retransmissions.

5.2 Accuracy Results

To evaluate how well the RATS algorithm addresses the Information Quality Tradeoff, we also present results highlighting the accuracy of the agents' network models, which depends on the quality of the information collected by the agents' sensing. For these results, we should expect the opposite of what we observed when evaluating the Performance Tradeoff: NA agents should provide the most accurate agent models because they only consider the Maximum Interval Bound, while PA agents do not consider *MAX_INT*, and RATS agents perform less average sensing than NA agents (c.f. Figure 4). However, this was not what we observed, as shown in Figure 5. Instead, RATS agents achieved the best overall accuracy (i.e., lowest error) in almost every level of resource contention, and NA actually performed worse than PA in most experiments. This result can be justified as follows. From our previous analysis and Figure 3, we know that NA agents experienced worse contention for network resources than the other sensing behaviors. Remembering our earlier assumption that the observer effect worsens as the contention for resources increases, we have an explanation for our results. Instead of providing higher quality information by waiting for a shorter interval between sensing, the increased contention in resource usage by NA agents caused a larger increase in the observer effect, decreasing the accuracy of the agents' observations. We can also see that the agents' accuracies all decreased after reaching larger contention (as indicated by experiments where average duration decreased in Figure 4), **verifying our previous assumption and the existence of the observer effect in multiagent sensing.** Thus, due to the observer effect, RATS agents better address the Information Quality Tradeoff than NA agents.

Between PA and RATS agents, we can observe that, in general, RATS agents achieved better accuracy by sensing at shorter inter-

vals. Thus, between PA and RATS agents, RATS sensing better balances the Information Quality Tradeoff by considering and adapting *MAX_INT*. However, one anomaly occurred at 65% external traffic, where PA agents were slightly more accurate. Considering Figure 3, we know that experiments with RATS agents were more congested at this level of resource contention. Thus, the benefit from shorter intervals between tracking in RATS was offset by an increase in the observer effect. Beyond 65%, however, PA agents also experienced more contention such that RATS was once again better than PA as expected. Thus, overall (except during experiments with the highest contention where all sensing behaviors performed poorly), our results imply that **the decrease in sensing caused by the Performance Tradeoff does not hurt the Information Quality Tradeoff, and RATS generally avoids decreases in accuracy caused by the observer effect.**

6. DISCUSSION

From our results, we have learned several valuable lessons. First, when comparing the impact of sensing on the environment (in terms of both average latency and simulation duration), RATS outperformed only-need-aware (NA) sensing by considering the Minimum Interval Bound to balance the Performance Tradeoff by avoiding too short of intervals between sensing, while also performing close to only-performance-aware (PA) sensing even with shorter average intervals (i.e., more sensing). Similarly, RATS outperformed all other sensing behaviors to achieve the highest model accuracy in almost every experiment by considering the Maximum Interval Bound to address the Information Quality Tradeoff, *until* experiencing decreases caused by resource contention and the observer effect, during which its accuracies were still close to the best achieved. Thus, RATS performs well at simultaneously balancing both the Performance Tradeoff between system performance and sensing, as well as the Information Quality Tradeoff between sensing quality and frequency when compared to only considering one tradeoff at once. Furthermore, by considering both tradeoffs simultaneously, RATS agents were able to gather more up-to-date information than PA agents and avoided large decreases in accuracy caused by the observer effect longer than NA agents. Thus, the RATS algorithm is the most capable of improving the relationship between agent sensing and limited resource environments among those evaluated in our study. Since the algorithm itself is rather simple and computationally efficient, without requiring communications between agents, it is well suited for application with highly constrained, rationally bounded agents (e.g., wireless sensor networks and robotics). However, the need to compare previous models (up to a specified *windowSize*) does increase the memory requirements for the algorithm.

Second, considering the better system performance resulting from sensing at lower levels of resource contention in Subsection 5.1, we have shed new light on the Performance Tradeoff. We observed that for resources where performance depends on how often the resource is consumed (e.g., PGM loss recovery), not only does the performance of the system decrease with too much resource usage, it also decreases with too little usage. Thus, the additional consumption resulting from agent sensing actually improved the system performance up to a point, after which performance began to deteriorate as expected. This dual-bounded nature is similar to the Information Quality Tradeoff, where too little sensing results in stale, out-of-date information, while too much sensing increases contention and can lead to decreases in quality due to the observer effect. Thus, for other limited resources whose performance depends similarly on usage (e.g., cache memory), we should expect similar improved performance from the RATS algorithm which attempts to balance the usage of the resource between two competing bounds.

Third, based on the performance and similarity between the various agent sensing behaviors in our experiments, we now better understand the impact of each sensing interval bound on both tradeoffs. This will help us design a metacognitive agent which is capable of weighting each bound to better fine-tune its performance with respect to both tradeoffs, depending on the current state of the environment. For example, we observed that when nearing heavy contention for resources (i.e., 65% external traffic), PA agents achieved better system performance and accuracy than both NA and RATS agents. Thus, if an agent can predict when it is in such a state (depending on its environment model), the agent could possibly improve its performance by reducing the influence of the Maximum Interval Bound to achieve performance more similar to PA agents. Before or after this state, the agent could instead rely on default RATS to achieve better sensing. Similarly, when system performance is not a concern, the metacognitive agent could also decrease the importance of the Minimum Interval Bound to possibly achieve even more accurate models without risking a decrease in system performance.

Finally, due to the large influence of the observer effect on sensing accuracy during periods of higher resource contention, there is an opportunity for research to both quantify this effect and consider it when reasoning about multiagent sensing and limited resource environments. For example, while our algorithm was able to achieve the coherent emergent behavior of improved system-wide resource moderation (as indicated by better accuracies and near optimal system performance with RATS), this moderation could probably be further improved through information and task sharing between cooperative agents. Each agent could share its experiences with similar agents to reduce the amount of sensing required in the environment. Such an environment could also provide a baseline for measuring the observer effect (when compared against active sensing by all agents), allowing the agents to quantify the observer effect and further decrease the impact of sensing on the environment. However, such an approach would also need to account for the effect of communication costs on both the Performance Tradeoff (communications are limited resources) and the Information Quality Tradeoff (large communication delays reduce the quality of data).

7. CONCLUSIONS AND FUTURE WORK

In conclusion, we have presented an adaptive, learning algorithm for Resource-Aware Tradeoff-based Sensing (RATS) to balance the tradeoffs between system performance and sensing due to resource consumption during sensing (*Performance Tradeoff*), and between sensing quality and frequency due to both the need for up-to-date information and the **observer effect** on measurements (*Information Quality Tradeoff*) **consuming the resource being sensed**. We conducted experiments to validate our algorithm against other adaptive sensing behaviors which only consider either of the two tradeoffs, and a baseline of no sensing. We discovered that RATS agents generally sense more accurate data than agents which only consider one of the tradeoffs, and RATS agents produce better system performance by considering the Performance Tradeoff. This implies that our computationally

inexpensive, no-communications algorithm is well suited for adapting sensing in limited resource, bounded rationality environments.

We also discovered various avenues for future work, which we plan to pursue. This includes improving the algorithm to share information and sensing tasks between agents to further minimize system-wide sensing and approximate the observer effect at the cost of additional communications. We will then conduct experiments comparing the new algorithm with RATS in different environments with varying communication costs. We would also like to incorporate the knowledge gained from these experiments about the effects (and side-effects) of each sensing behavior considered to build a metacognitive agent capable of adjusting its behavior between the three strategies depending on the perceived environment state to further improve both sensing accuracy and system performance.

8. ACKNOWLEDGEMENTS

This research was supported in part by a grant from the Department of Education (grant P200A040150), an NSF grant (DBI0742783), a Microsoft Research grant, and a UCARE grant through Pepsi and the University of Nebraska-Lincoln.

9. REFERENCES

[1] Anderson, M.L. and Oates, T. 2007. A review of recent research in metareasoning and metalearning, *AI Magazine*, 28(1), pp. 7-16.

[2] Avrahami, D. and Hudson, S.E. 2006. Responsiveness in instant messaging: predictive models supporting interpersonal communication. *Proc. CHI'06*, Montreal, Quebec, Canada, April 22-27, 2006, pp. 731-740.

[3] Breitgand, D., Cohen, R., Nahir, A., and Raz, D. 2007. Using the right amount of monitoring in adaptive load sharing. *Proc. ICAC'07*. Jacksonville, FL, June 11-15, 2007.

[4] Bull, S. and Greer, J. 2000. Peer help for problem-based learning, in *Proc. ICCE/ICAI'00*, 2, 2000, pp. 1007-1015

[5] Cao, J., McNeill, K.M., Zhang, D., and Nunamaker, J.F. Jr. 2004. An overview of network-aware applications for mobile multimedia delivery. *Proc. HICSS'04*, Big Island, HI, Jan. 5-8, 2004.

[6] Caripe, W., Cybenko, G., Moizumi, K., and Gray, R 1998. Network awareness and mobile agent systems, *IEEE Comm. Mag.*, 36(7), pp. 44 – 49.

[7] Chevaleyre, Y., Dunne, P.E., Endriss, U., Lang, J., Lemaître, M., Maudet, N., Padget, J., Phelps, S., Rodríguez-Aguilar, J.A., and Sousa, P. 2006. Issues in multiagent resource allocation. *Informatica*. 30, pp. 3-31.

[8] Ding, Y., Malaka, R., Kray, C., and Schillo, M. 2001. RAJA – a resource-adaptive java agent infrastructure. *Proc. AGENTS'01*. Montreal, Quebec, Canada, May 28 – June 01, 2001. pp. 332-339.

[9] Eck, A., Soh, L-K., Jiang, H., and Chou, T. 2007. Testing collaborative traffic over wireless protocols, in *Proc. FIE '07*, Milwaukee, WI, Oct. 10-13, 2007, pp. F4J-11 - F4J-16.

[10] Fujisawa, K., Hayakawa, S., Aoki, T., Suzuki, T., and Okuma, S. 2000. Real-time decision making for autonomous mobile robot using evolution strategy and anytime sensing. *Proc. IECON'00*, pp. 2809-2814.

[11] Heisenberg, W. 1927. Über den anschaulichen Inhalt der quantentheoretischen Kinematik und Mechanik. *ZP 43*, Mar. 1927. pp. 172--198. Reference from http://www.aip.org/history/heisenberg/bibliography/1920-29.htm

[12] Horvitz,, E.J. 1987. Reasoning about beliefs and actions under computational resource constraints. *Proc. 3rd Workshop on UAI*, Seattle, WA, July, 1987.

[13] Landeldt, B., Sookavantana, P., and Seneviratne, A. 2000. The case for a hybrid passive/active network monitoring scheme in the wireless internet. *Proc. ICON'00*. Sept. 5-8, 2000. pp. 139-143.

[14] Network Working Group. 2001. "PGM Reliable Transport Protocol Specification", RFC 3208, Dec. 2001.

[15] Nguyen, G.T., Katz, R.H., Nobel, B., and Satyanarayanan, M. 1996. A trace-based approach for modeling wireless channel behavior, in *Proc. 1996 Winter Simulation Conf.*, ed. J.M. Charnes, D.J. Morrice, D.T. Brunner, and J.J. Swain, Coronado, CA, Dec. 8-11, 1996. pp. 597-604.

[16] North, M.J., Collier, N.T., and Vos, J.R. 2006. Experiences creating three implementations of the Repast Agent Modeling Toolkit, in *ACM Trans. Modeling & Computer Sim.*, 16(1), pp. 1-25.

[17] Padhy, P., Dash, R.K., Martinez, K., and Jennings, N.R. 2006. A utility-based sensing and communication model for a glacial sensor network. *Proc AAMAS'06*. Hakodate, Japan, May 8-12, 2006, pp. 1353-1360.

[18] Raja, A. and Lesser, V. 2007. A framework for meta-level control in multi-agent systems. *JAAMAS*, 15, pp. 147–196.

[19] Sarne, D., Grosz, B.J. 2007. Sharing experiences to learn user characteristics in dynamic environments with sparse data. *Proc. AAMAS'07*, Honolulu, HI, USA, May 14-18, 2007, pp. 202-209.

[20] Soh, L-K., Khandaker, N., Liu, X. and Jiang, H. 2006. Computer-supported cooperative learning system with multiagent intelligence. *Proc. AAMAS'06*. Hakodate, Japan, May 8-12, 2006, pp. 1556-1563.

[21] Wei, W. and Chen, G. 2008. JavaStatSoft: design patterns and features. *Comp. Stats.*, 23(2). pp. 235-251.

[22] Wilcoxon, F. 1945. Individual comparisons by ranking methods., *Biometrics Bulletin*, 1(6), pp. 80-83.

[23] Tay, Y.C., and Chua, K.C. 2001. A capacity analysis for the IEEE 802.11 MAC protocol, *Wireless Networks*, 7(1), pp. 159-171.

[24] Zilberstein, S. and Russell, S.J. 1993. Anytime sensing, planning, and action: a practical model for robot control. *Proc. IJCAI'93*, Chambery, France, 2003. pp. 1402-1407.