Maintaining Shared Belief in a Large Multiagent Team

Prasanna Velagapudi*, Oleg Prokopyev+, Katia Sycara* and Paul Scerri*
* School of Computer Science, Carnegie Mellon University
+ Department of Industrial Engineering, University of Pittsburgh

***Abstract*—A cooperative team's performance strongly depends on the view that the team has of the environment in which it operates. In a team with many autonomous vehicles and many sensors, there is a large volume of information available from which to create that view. However, typically communication bandwidth limitations prevent all sensor readings being shared with all other team members. This paper presents three policies for sharing information in a large team that balance the value of information against communication costs. Analytical and empirical evidence of their effectiveness is provided. The results show that using some easily obtainable probabilistic information about the team dramatically improves overall belief sharing performance. Specifically, by collectively estimating the value of a piece of information, the team can make most efficient use of its communication resources.**

I. Introduction

Emerging and envisioned systems involve hundreds or thousands of autonomous systems cooperating in an environment to achieve complex joint objectives [8, 3]. A key to the individual and joint effectiveness of the team is for each member of the team to have accurate information about objects in the environment and any uncertainty associated with those objects [7]. Respecting communication constraints, the team must ensure, in a distributed fashion, that each member has the information it requires to perform its role in the team's mission. Algorithms to achieve this goal are critical to the success of teams of autonomous systems for applications ranging from tracking [8] to force protection [3] to searching for lost hikers [4].

Previous work has looked at various approaches to the problem of maintaining shared belief using techniques ranging from particle filters [9] to Bayesian filters [5] to decision theory [10]. However, previous work suffers from two key limitations. First, it does not scale simultaneously in the size of the shared state space and the size of the team. For example, sensor networks can be large, but their view of the environment is simple [6], while some UAVs have a complex view of the world, but there are only a few of them [5]. Second, previous algorithms focus on ensuring *every* member of the team has the most accurate view of the world. In practice, this is not necessary and, thus, unnecessarily taxes communication resources. For example, a UAV at one end of a large space will likely need only approximate information about the environment at the other end in order to be highly effective (or even optimal).

In this work, individual sensor readings are sent from agent to agent, as opposed to much previous work which forwarded compressed beliefs [9]. This mitigates problems associated receiving the same information multiple times and allows information to be forwarded anonymously, which is useful in a large team [5]. When an agent takes a reading with a sensor, it looks at the information gain due to the sensor reading to determine whether to share it. If it decides to share it, the reading is encapsulated in a *token* along with a pseudo-unique ID that distinguishes it from other potentially identical sensor readings. The token is then forwarded to a teammate, who integrates the sensor reading with their own beliefs. Depending on the information gain the receiving team member gets from that sensor reading, it can either pass the token on or stop propagation. This basic technique effectively leads to multiple members of the team jointly determining whether or not to widely propagate some information, reducing the need for any single agent to be precisely correct in estimating the importance of the sensor reading to the team.

This token-based algorithm will be effective if tokens are delivered to team members who gain information from the sensor reading on the token. Thus, a policy for propagating a token around the team has two components: (1) determining whether to further propagate the token and (2) to whom to send the token. This paper looks at three policies for determining whether to propagate a token further: a constant "time-to-live" propagation policy; one that allows agents to increase or decrease a token's "time-to-live"; and one that uses joint estimates of the sensor reading's value to the team to determine a propagation distance. In this work, we make the assumption that each agent has no specific information about any other agent, hence the only thing that can be done for (2) is to send the token randomly.

Analysis of the first two policies establishes performance expectation as well as probabilistically bounding the expected performance in limited cases. Empirical results support the analysis, while also illustrating the effectiveness of the approach. Specifically, the policy based on an estimate of value to the team performs at least as well as the other policies, mimicking the performance of other policies in regions where their policy performs well. When the agents receive a large number of sensor readings with widely varying value, this policy clearly outperforms the other policies.

II. Problem Statement

This section formally describes the problem addressed by this paper. Agents $A = \{a_1, \ldots, a_m\}$ are a team with a joint objective in a partially observable domain. Decisions about actions by the agents are based on state variables $X(t) = \{x_1(t), \ldots, x_n(t)\}$. The agents have uncertain sensors, thus via some filter they must determine the probability of each of the state variables. Agent a_i's probability distribution over X at time t is $P^i(X(t), t)$. Communication between agents is assumed to be point-to-point, with agents able to communicate directly with a known subset of team mates at any time.

The performance of the team will be adversely affected whenever its estimate of the state of environment varies from the actual state of the environment. The information difference (KL-divergence or similar) is $\Delta^i(X, P^i(X(t), t))$. The bigger $\Delta^i(\bullet)$, the higher the divergence. However, depending on their current activities, individual agents will not be equally effected by divergence. In general, they will only need to know precisely some values, while others can be coarsely understood or not known at all. Specifically, the cost of $\delta^i(\bullet)$ divergence to an agent a_i at a particular time is: $c(a_i, \delta^i(\bullet)) \rightarrow \mathcal{R}$. Define $C(a_i, \Delta^i(\bullet)) \rightarrow \mathcal{R}$ to be a sum over c.

Using their sensors, agents take sensor readings $s \in S$. A sensor reading influences $P^i(X(t), t)$ via some filter ϕ, $P^{i'}(X(t), t) = \phi(P^i(X(t), t), s)$. The only assumption made about the filter is that it is always better to have more sensor readings. Using the cost of information divergence and filter equations, the value of that sensor reading to a_i is:

$$\hat{v}(s, a_i) = C(a_i, \Delta^i(X, P^i(X(t), t)), t) -$$

$$-C(a_i, \Delta^{i'}(X, P^i(X(t), t)), t),$$

i.e., the change in cost. We assume $\hat{v}(s, a) \geq 0$. The value of s to the whole team is:

$$\hat{V}(s) = \sum_{a \in A} \hat{v}(s, a)$$

To share a sensor reading s, an agent creates a *token* t encapsulating s. The token includes a psuedo-unique ID and may contain additional control information pertaining to the propagation of s, as long as this information is of a reasonable and strictly limited size. The agent then transmits token t to a randomly selected agent.

An agent receiving a token t containing a sensor reading s retains a history that it can use to identify repeat visits of t. Practically, this history need only be for a short temporal window, as outdated sensor readings will disappear when their corresponding tokens cease to propagate. We assume that integrating a sensor reading twice has no effect, thus, $\hat{v}(s, a) = 0$ after an agent has received s. After it integrates s, it must determine whether to propagate t further, and if so, transmits it randomly.

We define a *unique* visit to be the first visit of a token containing a sensor reading s to an agent a. We also define an *interested* agent for a given system state to be one with $\hat{v}(s, a) > \epsilon$ in that state. In the remainder of this paper, we assume $\epsilon = 0$.

The optimal behavior is to share s with as many agents in A_s as possible while minimizing the number of communications needed. Clearly, these two goals are inversely related: the best case for gaining value is for a token to visit each agent at least once, which may require large numbers of visits because tokens are constrained to move randomly, and the best case for reducing visits is for agents never to send anything, which gains no value. Thus, the behavior of the token policy must be balanced between these two goals, such that both are met sufficiently for the domain-specific task.

Suppose we have $\hat{v}(s, a) > 0$ for $a \in A_s \subseteq A$ and $\hat{v}(s, a) = 0$ for $a \in A \setminus A_s$, i.e., A_s is the subset of agents interested in sensor reading s. Let A_v be the subset of agents visited by token t containing sensor reading s in a particular case. We can reformulate our problem as the optimization of two objective functions.

First, we have the proportion of $\hat{V}(s)$ attained:

$$f_v = \frac{\sum_{a \in A_v} \hat{v}(s, a)}{\hat{V}(s)} \quad (1)$$

Second, given a total number of visits N ($N \geq 1$), we have the proportion of *necessary communications*:

$$f_s = \frac{u_N^s - 1}{N}, \quad (2)$$

where u_N^s is the number of agents with $\hat{v}(s, a) > 0$ (i.e., $a \in A_s$) who have seen the token t, and thus $u_N^s - 1$ corresponds to *the number of unique visits to agents in A with $\hat{v}(s, a) > 0$*. We assume that the agent that creates the token is interested, and subtract one to correct for this initial agent obtaining the token without it making any visits.

The idea behind definition of u_N^s and f_s is straightforward. We wish to penalize unnecessary visits, so our metric must decrease as the number of visits increases. However, we should not penalize our desired behavior, visits to agents with $\hat{v}(s, a) > 0$. By using u_N^s, our metric will remain near one for any number of visits, as long as those visits consistently lead to agents for which $\hat{v}(s, a) > 0$.

Clearly, $f_v \leq 1$ and $f_s \leq 1$. In an ideal scenario for gaining value, in which the token visits all of the interested agents in the team and the team gains all possible value, we can see that $f_v = 1$. Similarly, in an ideal scenario for reducing visits, in which the token only visits interested ($v(s, a) > 0$) agents, $f_s = 1$. Thus, the overall goal of any token policy is to maximize f_v and f_s.

III. Approach

In this section, we evaluate the performance of three policies for token propagation. The first is a simple constant "time-to-live" (TTL) policy, referred to as the *C-policy*, which is the norm for most packet-based routing schemes. We define TTL as the number of visits remaining before the propagation of a token stops. The second is a simple linear policy, referred to as the *S-policy*, which increases the TTL by a constant whenever

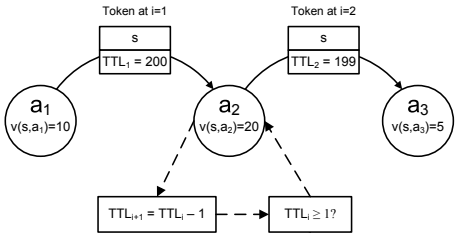

Fig. 1. C policy: Constant TTL policy, $c_0 = 200$. Agent a_1 generates a token t, sets $t.TTL = c_0$, then passes it to a_2. Agent a_2 decrements $t.TTL$, verifies that $t.TTL \geq 1$, and passes it to a_3.

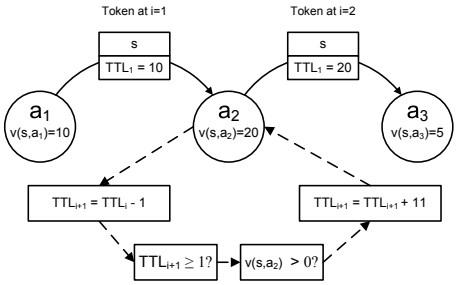

Fig. 2. S policy: Linear TTL policy, $c_0 = 10$, $c = 11$. Agent a_1 generates a token t, sets $t.TTL = c_0$, then passes it to a_2. Agent a_2 verifies that $t.TTL \geq 1$, and checks if the sensor reading s has value to itself. It does, so $t.TTL$ is incremented by c and then passed to a_3.

a token reaches an interested agent. Finally, we present an optimization based policy, referred to as the *S-OPT-policy*, that maximizes our objective functions directly at each agent that receives the token.

A. C-policy (Constant TTL policy)

First, we describe the most common and basic policy, a constant TTL. For every sensor reading s, the C-policy initializes the token $t = \{s, TTL = c_0\}$, then passes the token randomly to another agent. An agent receiving the token does the following:

1) Set $t.TTL = t.TTL - 1$.
2) If $t.TTL = 0$ stop the transmission of the token t.
3) Otherwise select another agent a uniformly randomly, and pass the token t to a.

An example of this policy is shown in Figure 1. The value of c_0 is the only parameter of the policy.

B. S-policy

Next, we describe a simple policy for adjusting the TTL of a token based on the value function $v(s, a)$. Now, each agent that the token visits will be able to modify the token TTL, with the intuition that this will allow agents to collectively determine an appropriate TTL for the token during its lifetime. This reduces the risk of an inaccuracy in a single agent's belief state leading to a poor communications decision that costs the entire team valuable bandwidth.

The S-policy begins by initializing the token in the same way as the C-policy, i.e., $t = \{s, TTL = c_0\}$. If the initializing agent is uninterested, the token is simply discarded. Otherwise, it is randomly propagated, and each agent a receiving the token does the following:

1) Set $t.TTL = t.TTL - 1$.
2) If $\hat{v}(s, a) > 0$ then set $t.TTL = t.TTL + c$.
3) If $t.TTL = 0$ stop the transmission of the token t.
4) Otherwise, select another agent a uniformly randomly, and pass t to a.

An example of this policy is presented in Figure 2. The values of c_0 and c are parameters of the policy.

C. S-OPT-policy

The third policy is directly derived from the objective functions in (1) and (2). The overall goal is to maximize the following weighted objective function, constructed from the linear combination of the original two objective functions of interest:

$$f = f_v - \gamma \frac{1}{f_s}, \tag{3}$$

where γ is a weighting parameter of the policy, which can be interpreted as a penalty for *unnecessary* visits. It can be observed that f in (3) will be maximized if we maximize f_v and f_s.

Initially, agent a_0 has some sensor reading s, which it has deemed worthy of transmission (we assume that $\hat{v}(s, a_0) > 0$). We create a token

$$t \quad = \quad \{s, \hat{V}_i^c(s), u_i^s, i\}, \tag{4}$$

where i is the total number of visits for token t (initially $i = 0$), $\hat{V}_i^c(s) = \sum_{a \in A_v} \hat{v}(s, a)$, i.e., the value received by the team after i visits and u_i^s is the number of *unique* visits to agents with $\hat{v}(s, a) > 0$, i.e., $a \in A_s$.

The S-OPT-Policy locally maximizes (3) at each agent a visited in order to incorporate the value $\hat{v}(s, a)$ and local estimates of $\hat{V}(s)$ and A_s. The policy starts by initializing the token, $t = \{s, \hat{V}_i^c(s), u_i^s, i\}$, with $t.i = 0$, $t.\hat{V}_0^c(s) = \hat{v}(s, a_0)$, $t.u_0^s = 1$. Then each agent receiving the token performs the following steps:

1) If $a \in A \backslash A_s$ or a was visited before, then $t.u_{i+1}^s = t.u_i^s$.
2) If $\hat{v}(s, a) > 0$ then set $t.u_{i+1}^s = t.u_i^s + 1$.
3) Set $t.\hat{V}_{i+1}^c(s) = t.\hat{V}_i^c(s) + \hat{v}(s, a)$.
4) Set $i = i + 1$ and analytically solve the following optimization problem (as described below):

$$\delta^* = \arg\max_{\delta} f = \frac{\hat{V}_i^c(s) + \hat{V}_\delta}{\tilde{V}_a} - \gamma \frac{i + \delta}{u_i^s + u_\delta^s}, \tag{5}$$

where $\tilde{V}_a$ is the estimation of $\hat{V}(s)$ by agent a, $\hat{V}_\delta$ is the expected value the team receives after an additional δ visits, u_δ^s is the expected number of visits to agents with $v(s, a) > 0$ (that is the number of unique visits to agents in A_s) during these δ visits.

In order to calculate u_δ^s, we need $\tilde{A}_s^a$, which is the estimation of A_s by agent a. Given these, the solution function is determined by analytically solving for a global maximum using $\frac{\partial f}{\partial \delta} = 0$. The resulting expression can be used to directly evaluate δ^* computationally. This

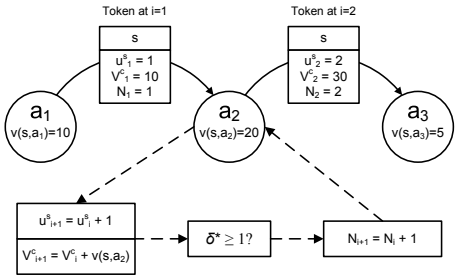

Fig. 3. S-OPT policy: Optimized TTL policy. Agent a_1 generates a token t, with the default initial values, adds its own value to V^c_1, then passes it to a_2. Agent a_2 updates the V^c_1 by adding its own value, and increments u^s because it has not received t before. It calculates δ^* and verifies that $\delta^* \geq 1$, so $t.N$ is incremented and then t is passed to a_3.

analytic solution need only be calculated once for given expressions of u^s_δ and $\hat{V}_\delta$.

5) Stop the token t if $\delta^* < 1$ (i.e., additional visits are expected to decrease the value of f).
6) Otherwise select another agent a uniformly randomly, and pass t to a.

An example of this policy is presented in Figure 3.

Observe that if $\gamma = 0$ (i.e., communication is not tightly constrained) then according to the S-OPT-Policy, the token will be transmitted as long as it is expected to gain value for the team. Conversely, if $\gamma = +\infty$ (i.e., communication resources are tightly constrained) then we can only continue the execution of the algorithm (i.e., the objective function in (5) will not be decreasing for $\delta \geq 1$) if we can guarantee that the next agent a has value $v(s,a) > 0$ with probability 1.

While gathering large proportions of value for the team, a token will revisit many agents before it is passed to new agents. This is purely due to the assumption that the token is randomly routed around the team and does not maintain any history. If any one of these visited agents underestimates the value of the token, the token's propagation may be stopped prematurely, because the policy may calculate the remaining agents or value in the system to be less than zero. In contrast, if an agent overestimates these values, there is little cost as subsequent agents will quickly stop propagation. Thus, slight inaccuracies in estimation are compounded when underestimating, but averaged out when overestimating. These problems are alleviated by bounding the estimation functions to be strict overestimate, i.e., $\tilde{V}_a \geq \hat{V}(s)$ and $\tilde{A}^a_s \geq A_s$.

IV. Analysis

In this section we provide some analysis of token propagation using the above described policies, specifically the C-policy and S-policy. This creates a framework for reasoning analytically about their performance. Furthermore, these expressions can be used to cross-verify later experimental results.

Since at each step of the described policies we select the next agent a uniformly randomly, we can calculate the expected number of interested agents, i.e., agents with $\hat{v}(s,a) > 0$, who have seen the token t after n visits:

$$u^s_n = (u^s_0 - |A_s|)\left(1 - \frac{1}{|A|}\right)^n + |A_s|,\ n \geq 0. \quad (6)$$

where u^s_0 is 1.

Therefore, we can estimate the expected proportion of necessary communications, f_s, as

$$E[f_s|N = n] = \frac{1}{n} \cdot (u^s_n - 1). \quad (7)$$

By our initial assumption, the agent $\tilde{a}$, which initiates the token passing, is interested, i.e., $v(s,\tilde{a}) > 0$, and this value is counted towards the total value that the team receives from the token t. Therefore, in the general case the expected value of $\sum v(s,a)$ the team receives during n visits should depend on the value $v(s,\tilde{a})$. Let us simplify the analysis by assuming that the agent that initiates token passing is selected uniformly randomly among agents in A_s. Let X_a be a random variable, which is equal to 1 if agent $a \in A_s$ has seen the token after n visits (i.e., its value $v(s,a)$ is collected by token t), and 0, otherwise. Then

$$E[f_v|N = n] = \frac{\sum_{a \in A} v(s,a) E[X_a|N = n]}{\hat{V}(s)} =$$

$$= \frac{u^s_n}{|A_s|} = 1 - \left(1 - \frac{1}{|A_s|}\right)\left(1 - \frac{1}{|A|}\right)^n \quad (8)$$

With these basic definitions in place, it is possible to directly evaluate the expectations of our optimization functions. First, we present these expressions for the C-policy.

Proposition 1 *Equations (7), (8) define performance of the C-policy for* $c_0 = n$.

Let us now consider the more interesting case of the S-policy. In order to estimate $E[f_v]$ and $E[f_s]$, we need to calculate $Pr\{N = n\}$, which is the probability that token t makes exactly n visits before its propagation is stopped by the algorithm.

Let c_0 and c be the parameters of the S-policy. Suppose after n visits the token visited i agents from A_s. Then the probability that during the next visit the token will encounter an unvisited agent $a \in A_s$ is equal to

$$p_i = \frac{|A_s| - i}{|A|}. \quad (9)$$

The probability that the token will get to a visited agent, or an uninterested agent from $A \setminus A_s$ is equal to

$$q_i = 1 - \frac{|A_s| - i}{|A|} \quad (10)$$

and, clearly,

$$p_i + q_i = 1.$$

The linear nature of our policy implies that termination only occurs in a very limited set of stopping states. For u unique interested agents visited, the token must stop after it

has exhausted the TTL accumulated by $u-1$ successful visits. Thus, the total number of visits must be exactly

$$N = c_0 + u + (u-1)(c-1), \tag{11}$$

where $u = 1, \ldots, |A_s|$. For example, at $u = 1$, the case where only one unique interested agent was visited, we have $N = c_0$. This is because we know that token t visited only 1 interested agent (the agent that initiated the transmission) and must have subsequently made exactly c_0 visits to uninterested agents in order for it to have stopped. Then in terms of (9) and (10) the probability that token t makes exactly n visits before its propagation is stopped can be calculated as follows:

$$\begin{aligned} &Pr\{N = n\} = \\ &= \begin{cases} 0, \forall u\ N \neq c_0 + u + (u-1)(c-1) \\ p_1 \ldots p_{u-1} \sum_{i_1,\ldots,i_u \in P_u} q_1^{i_1} \ldots q_u^{i_u}, \text{otherwise} \end{cases} \end{aligned} \tag{12}$$

where set P_u describes all possible paths to a stopping state with exactly N visits:

$$\begin{aligned} P_u = \{i_1, \ldots, i_u : i_1 + \ldots + i_u = c_0 + (u-1)(c-1), \\ \forall j \sum_{k=1}^{j} i_j < c_0 + (j-1)(c-1),\ j = 1 \ldots (u-1)\} \end{aligned}$$

Therefore, for any values of parameters c_0 and c we can estimate the performance of the S-policy in terms of f_v and f_s using (7)-(12) as follows:

$$E[f_v] = \sum_{u=1}^{|A_s|} \frac{u}{|A_s|} p_1 \ldots p_{u-1} \sum_{i_1,\ldots,i_u \in P_u} q_1^{i_1} \ldots q_u^{i_u}, \tag{13}$$

$$E[f_s] = \sum_{u=1}^{|A_s|} \frac{u-1}{c_u} p_1 \ldots p_{u-1} \sum_{i_1,\ldots,i_u \in P_u} q_1^{i_1} \ldots q_u^{i_u}, \tag{14}$$

where $c_u = c_0 + u + (u-1)(c-1)$.

We can also estimate the expected number of total visits:

$$E[N] = \sum_{u=1}^{|A_s|} c_u p_1 \ldots p_{u-1} \sum_{i_1,\ldots,i_u \in P_u} q_1^{i_1} \ldots q_u^{i_u}, \tag{15}$$

Proposition 2 *Equations (13), (14), (15) define performance of the S-policy.*

Probabilistic bounds on performance. We can also estimate the probability of scenarios when the token visits only a small portion ϵ of interested agents from A_s, that is:

$$Pr\{\sum_{a \in A_s} X_a \leq \epsilon \cdot |A_s|\}, \tag{16}$$

where $0 \leq \epsilon \leq 1$. Then we can calculate this probability as

$$Pr\{\frac{\sum_{a \in A_s} X_a}{|A_s|} \leq \epsilon\} = \tag{17}$$

$$= \sum_{u=1}^{N_\epsilon} p_1 \ldots p_{u-1} \sum_{i_1,\ldots,i_u \in P_u} q_1^{i_1} \ldots q_u^{i_u}, \tag{18}$$

(a) Network Parameters		(b) Policy Parameters		
Variable	Value	Policy	Variable	Value
A	500	C	c_0	2000
A_s	$0.5 \cdot A$	S	c_0	10
σ_V	$0.1 \cdot V(s)$	S	c	10
σ_A	$0.1 \cdot A$	$S - OPT$	γ	0.0464

TABLE I
DEFAULT PARAMETERS OF THE NETWORK MODEL.

where $N_\epsilon = \lceil \epsilon \cdot |A_s| \rceil$ is the maximum possible total number of the visited agents with $v(s,a) > 0$.

If we assume that $v(s,a)$ is a 0–1 function defined as

$$v(s,a) \quad = \quad \begin{cases} 1, a \in A_s \\ 0, \text{otherwise}, \end{cases} \tag{19}$$

then equation (18) also defines the proportion of the total value $\hat{V}(s)$ that the team receives from the token, since in this case:

$$Pr\{f_v \leq \epsilon\} = Pr\{\frac{\sum_{a \in A_s} X_a}{|A_s|} \leq \epsilon\}. \tag{20}$$

Knowing the values of parameters c_0, c, and $|A_s|$, we can now estimate the probability of scenarios when the token visits only a small portion ϵ of interested agents from A_s using equation (18).

V. RESULTS

The three policies described above were evaluated in an abstracted simulation. The simulation model used a fully connected network of A agents, of which A_s have $\hat{v}(s,a) > 0$. In each run, a token is spawned at an interested agent ($\hat{v}(s,a) > 0$) chosen randomly from the subset of interested agents in the team. The token is propagated according to a given policy, passed to agents selected randomly from the network. As each agent is visited, its value is set to zero, reflecting the assumption that the team will not gain any further value by having the token revisit the agent. The token continues to propagate until the policy determines it should be stopped, at which point the total number of visits, N, the proportional value gained by the agents visited, f_v, and the number of interested agents reached, u_N^s are used to evaluate the performance of the run.

Multiple parameters of the model are varied to simulate various effects in the system, but the nominal values for the model are provided in Table I(a). Using the simulation, it is straightforward to confirm the results of the analysis on the S-policy. For this, we consider a reduced team of size $A = 50$. Within this team, $|A_s|$ is varied between 1 and 50. For each agent $a \in A_s$, $v(s,a) = \mathcal{N}(100, 20)$. Each value of A_s is simulated for 20 runs, and the mean of each batch of runs is plotted in Figure 4 for N, f_s and f_v. Even with the high amount of variation introduced by the Gaussian value function, it is clear that the expectation functions are consistent with the simulated results.

With this analytic knowledge of the behavior of the two simpler policies, we can now begin to empirically gauge the

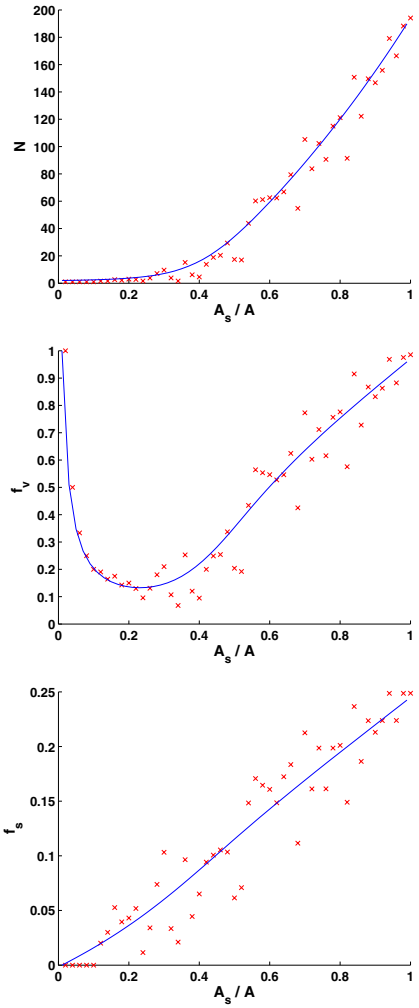

Fig. 4. Averaged simulation results compared to expected results for S-policy

quality of the third policy. The nature of the S-OPT policy requires that agents be able to strictly overestimate the total value and total number of interested agents in the system. Thus, the absolute value of 10% Gaussian noise was added to the actual values of the system to create strictly overestimating functions:

$$\tilde{V}(s) = V(s) + |\mathcal{N}(0, \sigma_V)| \quad (21)$$

$$\tilde{A}(s) = A(s) + |\mathcal{N}(0, \sigma_A)| \quad (22)$$

In order to deal with the various parameterizations of each policy, a baseline simulation was run using the nominal network settings while varying the tuning parameters of each policy. A rough scaling of parameters was obtained that corresponded to similar performance in each policy for the nominal network in terms of f_v and N. It was found that a linear scaling of c_0 (for the C-policy), c (for the S-policy) and logarithmic scaling of γ (for the S-OPT-policy) satisfy the necessary requirements. The other parameter (c_0 for the S-policy) was set to a constant value. These ranges are detailed in Figure 5.

Interestingly, when evaluated by proportional value collected, the C-policy follows a roughly logarithmic path, the S-policy follows a sigmoid path, and the S-OPT-policy appears to form a piecewise combination of the two paths, starting out with a logarithmic path, then switching to a sigmoid at a critical point. This hints that the underlying behavior of the S-OPT-policy is emulation of the most useful simple policy in a given region.

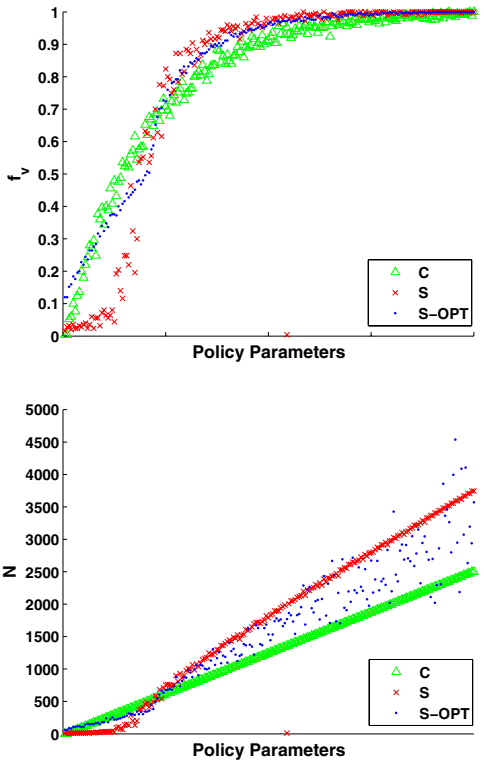

Fig. 5. Varying policy parameters. Along the x-axis, γ varies logarithmically between $10^{-0.1}$ and $10^{-2.5}$, c varies linearly between 0 and 15, and c_0 varies linearly between 0 and 2500.

Using this baseline, a set of parameters can be chosen that yield similar performance in each metric. The parameters used here are recorded in Table I(b).

With these parameters held constant, the network size was varied from 10 to 5000 agents. The results are presented in Figure 6. As the number of agents is increased, moving right along the x-axis, the effect on the C-policy is significant, dropping along f_v. This is an intuitive result: the number of visits stays constant, meaning that as the team gets larger, the value gained by the team passing a token does not scale along with the increase in value in the team. The S-policy and S-OPT-policy perform much better, increasing their number of visits linearly as the network size increases, and thus maintaining the proportional value collected. This is because these policies continue to see value in token propagation, while the C-policy does not take the context into account.

Next, the proportion of interested agents (A_s/A) in the network was varied between 0 - 100%, with the results in Figure 7. When the proportion of interested agents in the

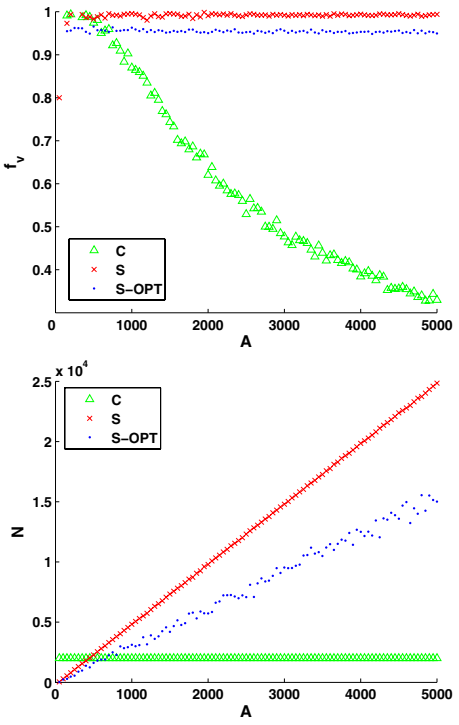

Fig. 6. Varying network size.

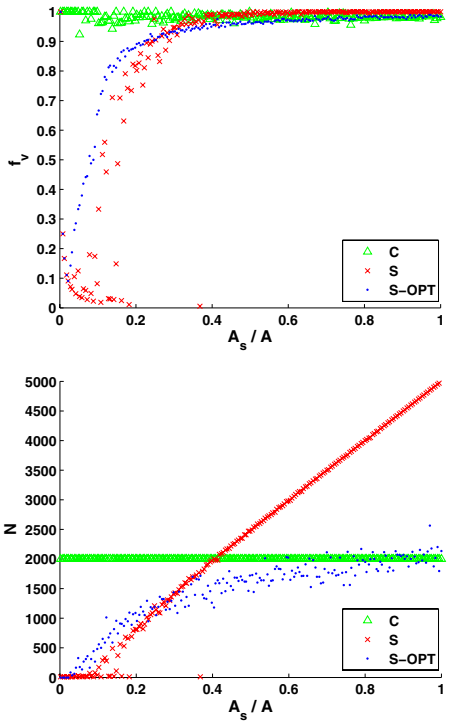

Fig. 7. Varying proportion of interested agents.

team is varied, the C-policy remains fairly constant in its effectiveness, as can be seen by the flat f_v over the entire range of (A_s/A). The S-policy, however, does not handle the variation as gracefully. At small values of (A_s/A), corresponding to only a few agents in the team being interested, it becomes prone to failure, dropping tokens prematurely if a few uninterested agents are visited sequentially. This is intuitive, as its linear increase in TTL depends finding a steady supply of interested agents. At higher proportions, corresponding to many agents in the team being interested, the S-policy overestimates its TTL. This causes a linear increase in the number of visits the token makes while not significantly increasing value collected. In fact, when the entire team is interested, the S-policy propagates the token 2.5 times farther than the other two policies, with fewer than a 5% gain in value. The S-OPT-policy mediates between the two policies, increasing TTL logarithmically such that when few agents are interested, a linearly proportional number of visits are made, but as the network grows more interested, an asymptotic number of visits are made such that a sufficient, but not unnecessary amount of value is collected by each token.

To understand the impact this has in an actual system, we set up a common scenario for multi-agent belief sharing. A_s for each token is sampled from a bimodal Gaussian distribution, with one peak at $\frac{1}{4}A$ and the other peak at $\frac{3}{4}A$. This represents a scenario where sensor readings are frequently of interest to a small portion of the team, but occasionally contain significant information that is of interest to a large portion of the team. By varying the amplitude of each peak in the distribution, as seen in Figure 8(a), we can adjust how often "important" sensor data is collected. The three policies were simulated over these distributions for a team size of 500, with 100 runs per policy, using the nominal parameters specified in Table I.

It can be clearly seen in Figure 8(b) that the three policies perform similarly in gaining value for the team. The largest change is in the S-policy, which improves very slightly, minimally surpassing S-OPT-policy. However, the corresponding graphs in Figure 8(c) reveal a large difference in the efficiency of the algorithms. Unsurprisingly, the C-policy remains constant. The S-policy sharply increases its number of visits as the high value peak of the A_s distribution becomes more prevalent, increasing by almost 50%. The S-OPT-policy, while remaining competitive in the value gained for the team, is able to use far fewer visits than either of the other policies in all cases. This includes the first case, where the S-OPT-policy is actually surpassing the S-policy in value gain as well.

VI. Related Work

There has been recent interest in the use of decentralized Bayesian filters such as the ones proposed in [5, 2] to manage beliefs over a large team. Communicating these beliefs, however, is expensive, prompting several selective communications approaches. Divergence metrics such as Hellinger affinity and KL-divergence are commonly used to measure the information gain of individual communications. However, existing methods of integration such as channel managers [1] or query-based particle filters [9] face scaling issues, in these cases, dealing with redundant data and polynomial-time scaling with team

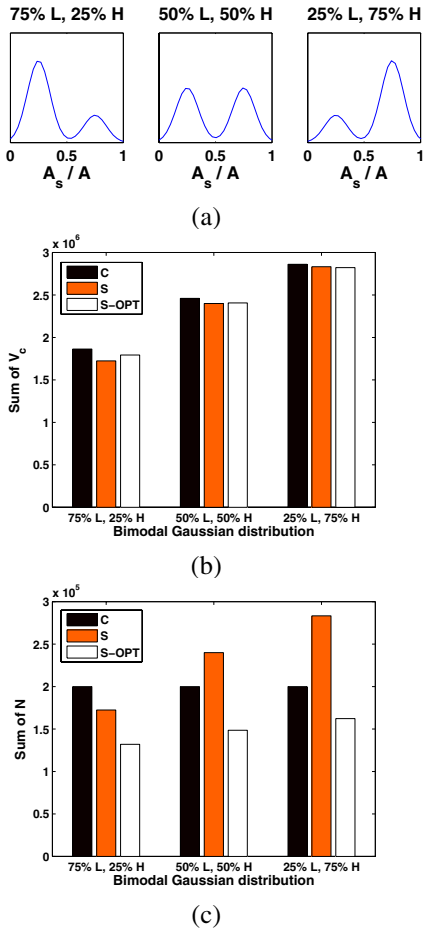

Fig. 8. Performance of policies with bimodal Gaussian distribution of A_s. Panel (a) depict the PDFs of the three bimodal test distributions. Panel (b) shows the cumulative total value collected by each policy over 100 runs. Panel (c) represents the cumulative total number of visits made by each policy over 100 runs.

size, respectively. Consensus protocols over communication networks are another method to achieve consistent belief among agents. Multi-hop relay protocols have been demonstrated to allow for fast consensus seeking [13], but previous work has focuses on trading robustness and convergence, rather than information gain and overall network traffic.

Token passing using random walks has been proposed in the field of peer-to-peer networking as a method of distributed search over large, unstructured networks, as it has been shown to approximate uniform random sampling [11]. Furthermore, token passing algorithms have recently been proposed as a suitable framework for large-scale team coordination [12]. In the area of communications, however, most this previous work has focused primarily on methods to improving routing of tokens, rather than on optimizing token lifetimes to reduce communications overhead.

VII. Conclusions

This paper presented a novel token-based approach to maintaining shared belief in a large team. Departing from previous work, it was not assumed that every agent needed an identical and best possible view of the environment. Relaxing this assumption allowed the development of a token-based belief sharing policy that used and refined an estimate of the value of a sensor reading to the team that was scalable and out-performed simpler policies. Analytic and empirical results identified the key properties of the approach.

Planned future work will make more increasingly realistic assumptions to make the algorithms described above more practically applicable. Such assumptions include assumptions about independence between sensor readings and Gaussian properties of value estimates. Additionally, we see that the key way to improve the practical efficiency of the algorithm is to find more intelligent ways of routing tokens around the network. Improved routing models will increase the complexity of analysis, likely requiring new analytic techniques. [1]

References

[1] F. Bourgault and H Durrant-Whyte. Communication in general decentralized filter and the coordinated search strategy. In *Proc. of FUSION'04*, 2004.

[2] F. Bourgault, T. Furukawa, and H Durrant-Whyte. Decentralized bayesian negotiation for cooperative search. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[3] L. Chaimowicz and V. Kumar. Aerial shepherds: Coordination among uavs and swarms of robots. In *7th International Symposium on Distributed Autonomous Robotic Systems*, 2004.

[4] J. L. Drury, J. Richer, N. Rackliffe, and M. A. Goodrich. Comparing situation awareness for two unmanned aerial vehicle human interface approaches. In *Proceedings of the IEEE International Workshop on Safety, Security and Rescue Robotics*, 2006.

[5] B. Grocholsky. *Information-Theoretic Control of Multiple Sensor Platforms*. PhD thesis, The University of Sydney, 2002. Available from http://www.acfr.usyd.edu.au.

[6] Andreas Krause, Carlos Guestrin, Anupam Gupta, and Jon Kleinberg. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Fifth International Conference on Information Processing in Sensor Networks*, 2006.

[7] A Makarenko, A Brooks, SB Williams, HF Durrant-Whyte, and B Grocholsky. An architecture for decentralized active sensor networks. In *IEEE International Conference on Robotics and Automation (ICRA'04)*, New Orleans, LA, USA, 2004.

[8] Charles L. Ortiz, Regis Vincent, and Benoit Morisset. Task inference and distributed task management in centibots robotic systems. In *AAMAS*, 2005.

[9] M. Rosencrantz, G. Gordon, and S. Thrun. Decentralized sensor fusion with distributed particle filters. In *Proceedings of the Conference on Uncertainty in AI (UAI)*, 2003.

[10] Milind Tambe. Agent architectures for flexible, practical teamwork. *National Conference on AI (AAAI97)*, pages 22–28, 1997.

[11] D Tsoumakos and N. Roussopoulos. Adaptive probabilistic search for peer-to-peer networks. In *Proc. Third International Conference on Peer-to-Peer Computing*, 2003.

[12] Y. Xu, P. Scerri, B. Yu, S. Okamoto, M. Lewis, and K. Sycara. An integrated token-based algorithm for scalable coordination. In *AAMAS'05*, 2005.

[13] Jin Zhipu. *Coordinated control for networked multi-agent systems*. PhD thesis, California Insitute of Technology, 2006.

[1]This research has been sponsored in part by AFOSR FA9550-07-1-0039, AFOSR FA9620-01-1-0542, L3-Communications (4500257512) and NSF ITR IIS-0205526