A Token-Based Approach to Sharing Beliefs in a Large Multiagent Team

Prasanna Velagapudi[1], Oleg Prokopyev[2], Paul Scerri[1], and Katia Sycara[1] ⋆

[1] The Robotics Institute, Carnegie Mellon University
Pittsburgh, PA 15213, USA
{pkv, pscerri, katia}+@cs.cmu.edu
[2] Department of Industrial Engineering, University of Pittsburgh
Pittsburgh, PA 15261, USA
prokopyev@engr.pitt.edu

Abstract. The performance of a cooperative team depends on the views that individual team members build of the environment in which they are operating. Teams with many vehicles and sensors generate a large amount of information from which to create those views. However, bandwidth limitations typically prevent exhaustive sharing of this information. As team size and information diversity grows, it becomes even harder to provide agents with needed information within bandwidth constraints, and it is impractical for members to maintain any detailed information for every team mate. Building on previous token-based algorithms, this chapter presents an approach for efficiently sharing information in large teams. The key distinction from previous work is that this approach models differences in how agents in the team value knowledge and certainty about features. By allowing the tokens passed through the network to passively estimate the value of certain types of information to regions of the network, it is possible to improve token routing through the use of local decision-theoretic models. We show that intelligent routing and stopping can increase the amount of locally useful information received by team members while making more efficient use of agents' communication resources.

1 Introduction

Emerging and envisioned systems involve hundreds or thousands of autonomous platforms cooperating in an environment to achieve complex joint objectives [1, 2]. These systems will generate incredible volumes of sensor data that need to be fused and disseminated to the platforms that need the produced information. In a distributed system, getting sensor data to platforms that require it in a timely manner while respecting tight communication constraints is a key problem. Algorithms to achieve this goal are critical to the success of teams of

⋆ This research has been sponsored in part by AFOSR FA9550-07-1-0039 and AFOSR FA9620-01-1-0542.

autonomous systems for applications ranging from tracking [1] to force protection [2] to searching for lost hikers [3].

The problem of distributed data fusion is a broad one, where many successful solutions have been developed for specific problem instances. The specific problem addressed in this chapter is characterized by three properties: (1) large numbers of agents generate a large volume of data about many features, (2) each agent gains different value from information about different features, e.g., an agent will be more interested in features in its spatial vicinity, and (3) the value an agent gains from information about a particular feature is not precisely known to the rest of the team in advance. While previous solutions have dealt with problems having one or two of these properties, using techniques ranging from particle filters [4] to Bayesian filters [5] to decision theory [6], there are no previous solutions that can handle distributed fusion problems with all three characteristics.

In many situations, especially in heterogeneous teams, agents will require information at varying levels of precision. In many approaches, this distinction is lost altogether, and information is distributed equally among teammates in this situation. However, this is highly inefficient when most of the agents do not need high precision about most of the features. For instance, a ground robot may require very extensive information about the position of nearby ground-level obstacles, while an aerial robot might need only rough estimates of large ground obstacles over a wide area. Notice that the agents that need particular information are not always the same as those that sense it. In this case, the aerial robot can use its perspective to get very detailed ground obstacle data over a wide area. Often, very few agents will attribute high value to precise knowledge of a particular feature, while equally few will be able to sense relevant information about that feature. For example, in a large team in a large outdoor environment, a few robots might be descending into a canyon. Information about the canyon, which might only be visible to a few robots exploring the cliffs above, is extremely valuable to this small subset. In situations like these, information must be shared *asymmetrically*, i.e., not all agents will receive the same information or even the same amount of information.

In this work, individual sensor readings are encapsulated into *tokens* and "pushed" from agent to agent. Each agent decides whether to continue to forward the token based on how useful to the team it believes the reading is. By encapsulating sensor readings and utilizing unique identifiers, we can altogether avoid the "double-counting" problem faced by other methods that condense or splice belief representations. In addition, sensor readings are usually compact, can be exchanged between agents with different filter algorithms, and are atomic and unordered.

In previous work [7], if the agent chose to forward the token, it did so randomly. In an asymmetric information environment this strategy is inefficient since, on average, all agents will receive the same information regardless of their need for that information. In this chapter, this inefficiency is addressed by building on the following two key ideas.

First, instead of considering the problem of *whether* to forward a token independent of *where* to forward it, we can combine both into a decision problem. Given a known cost of communication, we can estimate the value to the team of either forwarding the token to some teammate or terminating it, and use decision-theoretic methods to determine the optimal action. Previously received tokens encapsulate observations that influence the probabilities for the decision problem, e.g., receiving an observation indicating that a token was not useful to the previous agent along with information about a particular feature may indicate that it would not be useful to send another reading of the same feature to that agent in the near future.

However, if an agent only considers the impact of its routing on neighbors, its routing decisions can significantly impair communications throughout the network by redirecting tokens to greedy, suboptimal paths. The second key idea is to avoid making these myopic decisions about if and where to route the token by requiring the agent that has the token to estimate the token's impact on agents in other parts of the network. Our key insight here is that the token itself is an ideal carrier and accumulator for this information. If each token captures information about the agents it visits, it might be possible for recipient agents to make better estimates of the value of that information to the team. This, in turn, has the potential to improve the routes that agents use for subsequent tokens. However, in a large team, it is impractical for tokens to carry individual information about each visited agent. Instead, we have the token itself contain the aggregated estimate of the value to the team along a given path, calculated and updated by each agent it visits. This way, the decisions made by an agent can be based on summarized network statistics, while the estimate itself involves only constant time computations at each agent. We demonstrate that these simple estimates can improve the routing of tokens through a small worlds network (in which each node has relatively few neighbors, but any pair of nodes can be connected by a short path [8]) while maintaining or reducing the number of communications required.

2 Problem Statement

This section formally describes the problem addressed by this chapter. 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 $X(t) = \{x_1(t), \ldots, x_n(t)\}$ that describe their environment. These state variables can have any mathematical type (e.g., discrete, continuous, boolean) as long as the following functions are defined appropriately.

Agents take readings with their sensors. These sensors are imperfect, thus agents must use a filter to estimate a distribution over each of the state variables. Agent a has a probability distribution over X at time t of $P_a(X(t), t)$. While agents need not be homogeneous, it is assumed that each agent's filter can handle sensor readings from any other agent's sensor. Communication between agents is assumed to be fixed cost and point-to-point, with agents able to communicate directly with a known, static subset of teammates at any time, which are referred to as *neighbors*. No assumptions are made about the spatial or informational association between agents, but the communications network as a whole is assumed to have a small world property [8]. Denote by N_a the set of neighbors with which agent a can communicate. Let κ be the cost of a single communication, and denote by M_t the cumulative number of communications made before time t.

The performance of the team will be adversely affected whenever its members' estimates of the state of environment differ from the actual state of the environment. The information difference of a single agent is $\varDelta^a(X, P_a(X(t), t))$ (e.g., Kullback-Leibler divergence, or a similar measure). The bigger $\varDelta^a(\bullet)$, the higher the value of the divergence. However, depending on their current activities, individual agents' performance 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 $\varDelta^a(\bullet)$ divergence to an agent a at a particular time is: $c(a, \varDelta^a(\bullet)) \rightarrow \mathcal{R}$. For example, if a ground robot were traveling quickly across rugged terrain, the cost of divergence for a state variable representing the position of a nearby obstacle might be quite high, since it could potentially endanger the robot. The same variable might have a low cost of divergence for an aerial robot, as it would pose no threat and would not affect the decisions that the robot needed to make.

As agents receive sensor readings, they are integrated into $P_a(X(t), t)$ via some filter ϕ, $P'_a(X(t), t) = \phi(P_a(X(t), t), s)$. 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 is

$$v(s, a) = c(a, \varDelta^a(X, P'_a(X(t), t))) - c(a, \varDelta^a(X, P_a(X(t), t))) \ ,$$

i.e., the change in cost. We assume $v(s, a) \geq 0$. In a situation where a team of robots were estimating the location of a landmark, this value would be high for a high precision readings that could significantly improve robots' estimates. In contrast, if the robots already knew the position of the landmark to a high degree of accuracy, a noisy sensor reading would change very little, and so it would map to a small value. The value of s to the whole team is

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

The objective is to pass sensor readings around the team such that the total cost to the team is minimized after communication costs are considered:

$$\min \sum_{a \in A} c(a, \varDelta^a(X, P_a(X(t)))) + \kappa \cdot M_t \ . \tag{1}$$

Although omitted in the above expression for clarity, this minimization covers the space of potential paths through the network of every sensor reading generated over the time interval. Note that agents cannot directly measure $\Delta^a(X, P_a(X(t), t))$. Instead, agents estimate $v(s, a)$ directly, using their filters and domain-specific knowledge of x. For example, aerial robots receiving ground obstacle information would realistically be able to determine that such information was of little value to them, regardless of the accuracy of their filter. Denote this estimation function as $\hat{v}(s, a)$.

3 Algorithm

An agent cannot directly observe the value gained by its neighbors for a given token. However, it can get observations of the value gained by previous tokens of the same type that previously visited its neighbors. There also exists some transition function that determines how the value at the neighbors decreases as they receive tokens from various sources. An agent a is then faced with choosing an action from the set of possible actions Ψ, defined as

$$\Psi = \left\{ \bigcup_{b \in N_a} \mathcal{A}_b \right\} \bigcup \mathcal{S}$$

where $\mathcal{A}_b$ is the action of forwarding the token to neighbor b, and $\mathcal{S}$ is the action of stopping the token.

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, if so, (2) to whom to send the token.

The problem of estimating future value from only passively observed token traffic is hard. For divergence-based value functions, the state transition function upon receiving a token is often non-linear. In addition, there is the problem of repeating visits. If an agent a receives a token from an agent b that has a history of high-value, it is likely that sending a token of the same type to b will also result in high value. However, a token that has already been sent to b will not gain much additional value, as it will likely be revisiting nodes. Thus, we also need to estimate the likelihood that agents in b's neighborhood have already received a token, and either converge to a tree-like structure or rapidly update value estimates to compensate.

Given these issues, it is impractical to try and create an exact value or transition model based on the actual movement of tokens. We instead create a local heuristic framework for solving this decision problem based on the following insights: (1) the average value that an agent gains from tokens of a particular type is distributed in a roughly Gaussian way, and the mean of this value can be reasonably approximated, (2) we can represent a near-optimal solution to this problem as a probability distribution over the discrete action space, (3) many tokens move through the network, so at least a few will travel over each link, and (4) while agent-token value will change over time, it will change slowly with respect to the token movement through the network.

We have tokens of various types $\Gamma = \{T_1, T_2, \ldots, T_{|\Gamma|}\}$. Each type of token contains data related to the corresponding state variable, i.e. T_1 contains data about x_1, T_2 about x_2, and so forth. A token τ of type $T \in \Gamma$ is defined as a tuple containing three elements: $\tau =< s, b, E >$, where s is the sensor reading about x_i, b is the previous agent visited, and $E \triangleq E^T_{ab}$ is the expected value of sending a token of type T from agent a to its neighbor b. This is computed every time a token is to be forwarded from an agent b to an agent a. We describe the details of this process below.

Each agent a maintains a decision matrix D_a, of dimension $|\Gamma| \times (|N_a| + 1)$ from which it samples its routing actions. Each column of the matrix represents the possible actions that agent a has for a token. Columns 1 to $|N_a|$ are the actions of routing to the neighbors of a, while column $|N_a| + 1$ is the action of stopping the token. Each row in the matrix represents a type of token from set Γ. Element $D_a[T, \psi]$ is the probability of executing an action $\psi \in \Psi$ when given a token of type $T \in \Gamma$. Thus, each row of D_a is a well-formed probability distribution, satisfying

$$\begin{aligned} D_a[T, \psi] &\geq 0 \quad \forall\psi \quad \forall T \\ \sum_{\psi} D_a[T, \psi] &= 1 \quad \forall T \ . \end{aligned}$$

Each agent also maintains a value matrix V_a of dimension $|\Gamma| \times (|N_a| + 1)$ that contains its value estimates for each potential routing action. Once again, each column of the matrix is an action, and each row denotes a type of token. Element $V_a[T, \psi]$ is the estimated value gained by the team if action $\psi \in \Psi$ is performed on a token of type $T \in \Gamma$.

When an agent a receives a token of type $T \in \Gamma$, it samples an action from the distribution in the respective row of D_a. If that action is to send to a neighbor b, then the agent computes the expected value estimate

$$E^T_{ba} = \hat{v}(s, a) + \sum_{c \in N_a \setminus \{b\}} \left(D_a[T, \mathcal{A}_c] \cdot V_a[T, \mathcal{A}_c] - \kappa \right) \ . \tag{2}$$

Intuitively, this is simply the expected value of sending a packet from agent b to agent a, using an expectation estimate that incorporates a split-horizon. This value is then stored in the token, and the selected action of forwarding is performed. If the stopping action is selected, the token is simply deleted. While this estimate does not include an explicit discount factor, the probabilities used in the expectation are later constrained to be strictly less than one. In practice, they effectively act as implicit discount factors.

3.1 Heuristic Updates of D_a and V_a

As tokens are received by an agent $a \in A$, D_a and V_a are updated based on the incoming value estimates. We define two heuristic functions that govern this behavior. First, a value update function λ is applied to incorporate the estimate contained in a received token τ into the value matrix V_a:

$$V_a \leftarrow \lambda(V_a, \tau.E) \ . \tag{3}$$

Then, a decision update function π uses the updated value matrix to refine the decision matrix D_a:

$$D_a \leftarrow \pi(D_a, V_a) \ . \tag{4}$$

Value estimation (λ). Each agent maintains an estimate of the value of each type of token, based on a simple adaptive learning rule. This estimate is calculated using a learning rule of the form

$$V_a[T, \mathcal{A}_b] \overset{\lambda}{\leftarrow} (1-\alpha) \cdot V_a[T, \mathcal{A}_b] + \alpha \cdot \tau.E \tag{5}$$

where α is the *value learning factor*, and $\tau.E$ is the expected value estimate in the received token τ. This will compute a weighted average of the new measurement and the current estimate. The larger the value of α, the more sensitive the value estimation to noise. If α is too small then value estimation will not respond fast enough to the system dynamics.

Routing (π). We define a heuristic for updating the decision matrix D_a that follows the intuition: *If routing action ψ is n-times more useful than action ψ', then action ψ should be n-times more likely to occur than action ψ'.* Here, "usefulness" is represented by the value estimate. This intuition is embodied in the simple update rule

$$w_{\psi}^{T} = V_a[T, \psi] + 1 - \min_{\phi \in \Psi} V_a[T, \phi] \tag{6}$$

$$D_a[T, \psi] \overset{\pi}{\leftarrow} (1-\epsilon) \cdot \frac{w_{\psi}^{T}}{\sum\limits_{\phi \in \Psi} w_{\phi}^{T}} + \frac{\epsilon}{|\Psi|} \tag{7}$$

where w_{ψ}^{T} is the weight of action ψ for a token of type T. Here, the minimum value is subtracted and an offset of one is added to project the estimated value $V_a[T, \psi]$ into the range $[1, +\infty)$. This conveniently allows the weight to be normalized to obtain probabilities in the row of D_a corresponding to T. A small constant factor ϵ is used to ensure that every route is selected with some non-zero probability, thus preventing situations where agents never receive tokens from a particular neighbor.

Combining the two update rules, we formulate the token handling procedure that each agent performs when receiving or generating a token. The resulting algorithm is summarized in Algorithm 1.

Algorithm 1 Handle token.

```
1: $\tau \leftarrow getToken()$
2: $\psi \sim D_a[\tau.T]$
3: if $(\psi \neq \mathcal{S})$ then
4:     Calculate $E_{ba}^{\tau.T}$ using Equation 2
5: end if
6: $V_a = \lambda(V_a, \tau.E)$
7: $D_a = \pi(D_a, V_a)$
8: if $(\psi \neq \mathcal{S})$ then
9:     $\tau.E = E_{ba}^{\tau.T}$
10:    $send(\tau)$
11: else
12:    $delete(\tau)$
13: end if
```

4 Experimental Results

In this section, we describe an empirical evaluation of our approach, which we refer to below as a *proportional routing* policy. A simulator was constructed to evaluate this policy in a scenario of 500 agents performing 1-D target tracking. This simple setup was sufficiently complex to test the basic dynamics of a large team without introducing irrelevant effects that could obfuscate the performance of the policy. The agents were connected by a small worlds communication network, with an average of about 7 communications links to other agents. 1% of the agents were equipped with simulated sensors that generated Gaussian estimates of the target position every 5 time steps, emulating a discrete sampling rate. Every agent maintained a local Kalman filter using a static motion model with high process noise. As there was only one state variable, we used only one type of token, i.e. $|\Gamma| = 1$. The target was initialized at a random Gaussian position, and proceeded to follow a constant velocity trajectory for the duration of the simulation. Each trial was run for 10^4 time steps. Results were averaged over 10 trials for each condition.

Agent a's need for an accurate estimate was represented by a weight C_a. The distribution of information need over the team was bimodal, with a small proportion of the agents having high need, $\mathcal{N}(10^6, 10^5)$, and the remainder having low need, $|\mathcal{N}(0, 1)|$. The objective was to minimize the weighted KL-divergence of the team at the end of the simulation, defined as the following sum:

$$WD = \sum_{a \in A} C_a \cdot \Delta^a(X, P_a(X(t), t))\bigg|_{t=10^4}$$

Note that this is the first term in the original cost function in Equation 1, applied using KL-divergence as the divergence metric. The value approximation function used by the agents was the covariance of the agent's filter multiplied by the information need constant. By separating this from the communications cost, it was possible to analyze the tradeoffs agents made between communications and value under the test conditions.

Two policies were tested, the proportional routing policy, and a random routing policy. The random policy simply routed each token uniformly randomly for a constant number of transmissions, then terminated it.

In the first experiment, the weighted divergence of the policies was studied as the percentage of high-C_a and sensor-equipped agents was simultaneously varied. For fair comparison, each trial of the proportional policy was matched with a trial of the random policy which was adjusted such that policies had the same average number of token communications. The resulting weighted divergence in Figure 1 shows that the proportional policy outperforms the random policy in situations where both few agents are high-C_a and few are producing the relevant sensor readings. Specifically, this was the case when the high-C_a and sensor-equipped percentages were below 5%.

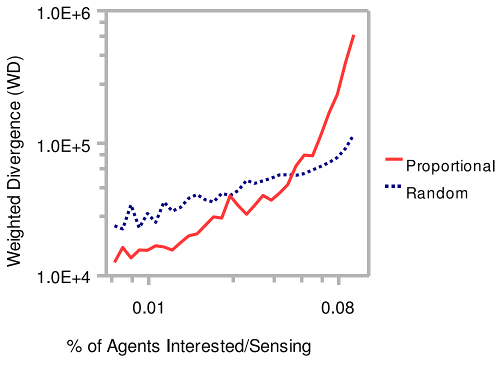


Fig. 1: A comparison of the random and proportional routing policies over changing interest levels.

As a baseline, the random policy was tested to determine the effects of average token transmissions on weighted divergence. Figure 2 shows that the random policy's weighted divergence drops rapidly as the average number of communications increases at small numbers of transmissions, but flattens asymptotically after about 10 transmissions. This suggests that there may not be much benefit to tokens having extremely long lifetimes, as they will not improve the weighted divergence any further.

Next, an experiment was done to evaluate the effects of the communications cost on the proportional policy. By varying the cost, the behavior of the policy can be adjusted to prioritize decreasing communications over increasing value. In Figure 3, communications cost proportionally impacts the average number of communications, while the relationship with weighted divergence is less evident. It is possible that the reduction of transmissions more directly impacts longer,

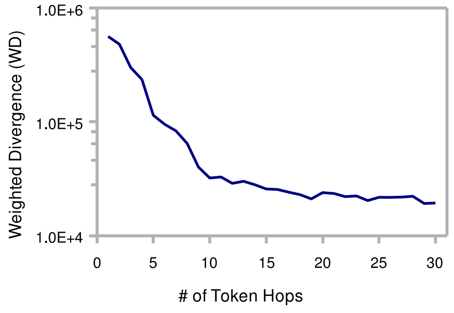


Fig. 2: The effects of increasing token lifetime while using the random policy.

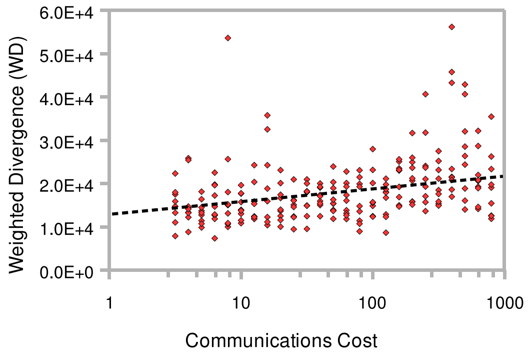
suboptimal routes before shorter, high-C_a ones, mitigating the effects of the cost increase. This is consistent with the previous observation that token lifetimes above some threshold provided diminishing returns in weighted divergence.

Closer examination of an individual trial, shown in Figure 4, provides insight into the behavior of the algorithm. Weighted divergence steadily increased during periods when high-C_a agents did not receive sensor readings, then dropped sharply when readings were delivered. The spacing and magnitude of these ramps suggest that the proportional routing policy reduces weighted divergence by being more consistent in its delivery of sensor readings to high-C_a agents.

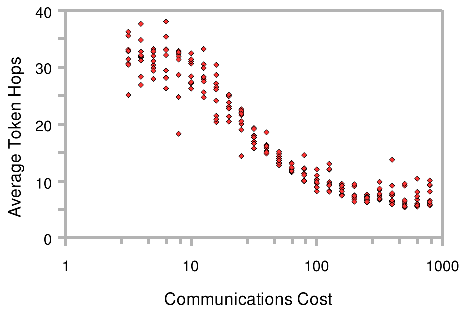
These experiments show that the proportional policy is relatively more effective than the random policy when low percentages of the team are producing and consuming sensor readings. This is a region where many current methods are inefficient, hence, this situation exemplifies the type of problem for which this algorithm was designed. The policy is also capable of dynamically adjusting the number of communications as cost increases, while maintaining effective routing. Finally, we see diminishing returns in value as token lifetimes are increasing, suggesting that reasonably short lifetimes are sufficient to maintain low weighted divergence over the team.

5 Related Work

Much previous work focuses on sending beliefs after filtering has occurred, which requires precautions to be taken to avoid “double-counting”, in which multiple filter updates include information derived from the same sensor reading. One frequent solution to this problem is imposing an acyclic structure over the network [9] [10], which introduces scalability and dynamics issues. In contrast, this work assumes updates are generated for individual sensor readings. By treating each reading atomically, hashing mechanisms can trivially handle the problem of receiving the same information multiple times by storing a history of token hashes and ignoring revisiting tokens, while still allowing information to be forwarded anonymously, which is useful in a large team [5]. These token histories

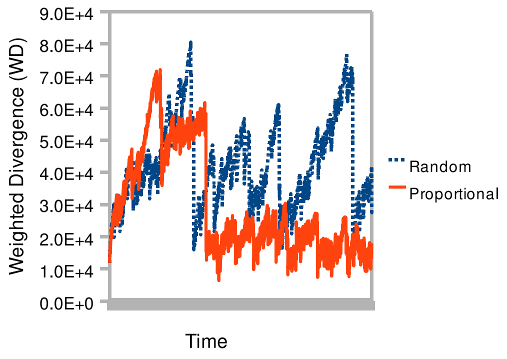


(a) Weighted divergence is logarithmically affected by cost. (some outliers are not visible)

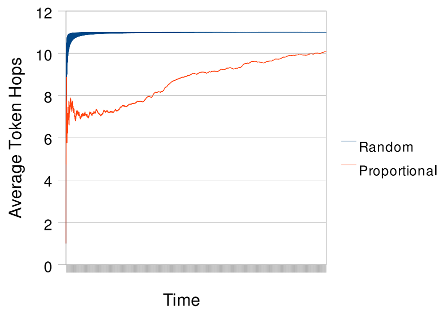


(b) Number of token transmissions decreases log-sigmoidally as cost increases.

Fig. 3: Proportional routing policy behavior over changing communications cost.



(a) The proportional policy begins routing to high-C_a agents to reduce divergence sharply.



(b) Both policies use the same total number of token transmissions, but the proportional policy uses its transmissions more conservatively.

Fig. 4: A comparison of the random and proportional routing policies over a single trial.

need only be temporary, as individual tokens have relatively short lifetimes. Note that individual readings could be replaced by any other atomic method of information sharing, such as exchanging particles between team members' particle filters [4].

Token-based methods have been shown to be effective in large-scale team coordination tasks, including the task of information sharing [11]. Using only information pushed forward in tokens, Xu demonstrates that adaptive probabilistic routing can improve team performance [12]. However, these previous approaches to information sharing do not explore token termination conditions, and require tokens or agents to store path histories over their lifetime. Other related methods include a dynamic optimization-based strategy for computing token lifetimes under assumptions of random routing with peer- to-peer communication [7], and an adaptive routing method that uses learning rules similar to Equation 5 to self-optimize routing over dynamic networks.

6 Conclusions

This chapter presented a novel approach to sharing information in large teams using tokens. In contrast to previous work, it was assumed that members of the team had vastly different needs for the information generated by other agents in the team. In our experiments, this need was concentrated among a small percentage of the team. Under these situations, this approach adjusts local routing and stopping probabilities to improve information sharing performance over the team. Empirical results demonstrate this efficiency in a simple simulated tracking problem despite the sparse information agents had with which to make routing decisions.

The experiments also show a number of interesting properties of this approach and the problem of information sharing in teams with asymmetric need. One surprise was the unexpectedly high performance of a random routing policy in reducing divergence, even in highly asymmetric situations. Analytically, it may be possible to bound the optimality of random routing in this problem, and use it as a baseline for comparing other techniques. In contrast, this approach seemed to lose efficiency when faced with large numbers of interested agents. However, hybrid approaches might be possible within this token framework that can use the proportional routing heuristic when routing information destined for a small subset of agents in a team, then switch to a different heuristic when routing more commonly desired information, all using the same estimation methods. If this framework can be extended to work across a wider range, it will provide a lightweight, dynamic approach to sharing information in teams with asymmetric information needs.

References

1. Ortiz, C.L., Vincent, R., Morisset, B.: Task inference and distributed task management in centibots robotic systems. In: AAMAS. (2005)

2. Chaimowicz, L., Kumar, V.: Aerial Shepherds: Coordination among UAVs and Swarms of Robots. In: 7th International Symposium on Distributed Autonomous Robotic Systems. (2004)
3. Drury, J.L., Richer, J., Rackliffe, N., Goodrich, M.A.: 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)
4. Rosencrantz, M., Gordon, G., Thrun, S.: Decentralized sensor fusion with distributed particle filters. In: Proceedings of the Conference on Uncertainty in AI (UAI). (2003)
5. Grocholsky, B.: Information-Theoretic Control of Multiple Sensor Platforms. PhD thesis, The University of Sydney (2002) Available from http://www.acfr.usyd.edu.au.
6. Tambe, M.: Agent architectures for flexible, practical teamwork. National Conference on AI (AAAI97) (1997) 22–28
7. Velagapudi, P., Prokopyev, O., Sycara, K., Scerri, P.: Maintaining shared belief in a large multiagent team. In: Proceedings of Tenth International Conference on Information Fusion. (2007)
8. Watts, D., Strogatz, S.: Collective dynamics of small world networks. Nature **393** (1998) 440–442
9. Makarenko, A., Brooks, A., Williams, S., Durrant-Whyte, H., Grocholsky, B.: An architecture for decentralized active sensor networks. In: IEEE International Conference on Robotics and Automation (ICRA'04), New Orleans, LA, USA (2004)
10. Nettleton, E., Thrun, S., Durrant-Whyte, H.: Decentralized slam with low-bandwidth communication for teams of airborne vehicles. In: Proc. of International Conference on Field and Service Robotics. (2003)
11. Xu, Y., Scerri, P., Yu, B., Okamoto, S., Lewis, M., Sycara, K.: An integrated token-based algorithm for scalable coordination. In: AAMAS'05. (2005)
12. Xu, Y., Lewis, M., Sycara, K., Scerri, P.: Information sharing in very large teams. In: AAMAS'04 Workshop on Challenges in Coordination of Large Scale MultiAgent Systems. (2004)