MAC Delay in Belief Consensus for Distributed Tracking

Christopher Lindberg*, L. Srikar Muppirisetty*, Karl-Magnus Dahlén†, Vladimir Savic‡, Henk Wymeersch*
*Department of Signals and Systems, Chalmers University of Technology,
Gothenburg, Sweden, e-mail: {chrlin,srikar.muppirisetty,henkw}@chalmers.se
†HiQ Consulting, Gothenburg, Sweden, e-mail: karl-magnus.dahlen@hiq.se
‡Department of Electrical Engineering, Linköping University, Linköping, Sweden, e-mail: vladimir.savic@liu.se

***Abstract*—In target tracking applications where many sensors must have a common view of the target's state, distributed particle filtering with belief consensus is an attractive solution. It allows for a fully distributed, scalable solution, guarantees consensus in connected networks, and converges fast for network with high connectivity. However, for medium access control, high connectivity is detrimental, possibly leading to a different convergence/performance trade-off. We study the delay/performance trade-off of distributed particle filtering with belief consensus in the presence of time division medium access control. We found that for small networks, (i) the impact of max-consensus should be accounted for; (ii) a simple round-robin schedule combined with a large communication range gives the best delay/performance trade-off.**

I. Introduction

An important application of wireless sensor networks (WSN) is to cooperatively track changes in an environment or of an object over time, by taking individual sensor's measurements and combining these to form a global estimate. This sensor fusion can be implemented in a fully distributed way to improve robustness to node and link failures, to achieve a high degree of scalability, to evenly distribute energy consumption in the network, and to ensure that a global estimate can be accessed from any sensor in the WSN [1].

Under certain conditions, distributed methods based on the Kalman filter can be applied [2]. However, due to non-linearities or non-Gaussian noise, algorithms based on particle filters are a popular choice for distributed tracking (see [1] for a recent overview). Such a distributed particle filter (DPF) generally utilizes some form of distributed belief consensus (BC) algorithm to ensure agreement among sensor nodes about the target's state. In [3], every sensor has the same set of particles at every time step (achieved through proper seeding of random number generators) and relies on average belief consensus to agree on the particle weights. In [4], a consensus based algorithm for distributed target tracking is combined with a cooperative self-localization method based on non-parametric belief propagation to improve the results of both tasks. Average consensus can also be applied to other parameters, such as the parameters of the posterior distributions [5], [6] or the likelihood functions [7], [8].

The convergence behavior of consensus algorithms with different practical impairments is well-studied in the literature. Example include quantization [9]–[11] and packet loss [12]. The convergence rate is generally studied by investigating the flow of information through the network, though little is known regarding the impact of packet collisions or medium access control (MAC) [1]. To the best of our knowledge, this issue is only studied in [13], where it was shown that for dense, two-dimensional networks, the convergence rate asymptotically no longer depends on the communication radius when scheduling is accounted for. However, it is not clear if this conclusion is valid for target tracking applications in small and medium-sized networks with specific performance requirements.

In this paper, we investigate how MAC delay and performance trade-off in a DPF with belief consensus. In particular, we consider different communication ranges, and form collision-free broadcast schedules. Each execution of such a schedule corresponds to one consensus iteration. This allows us to determine the time to execute a certain number of consensus iterations. In combination with the performance degradation with respect to a centralized solution, we are able to determine the time required to achieve a certain required performance.

II. System Model

A. Sensor and Target Model

We consider a WSN deployed in a surveillance area. The WSN consists of N_s static sensors with known positions, where the position of the n-th sensor is denoted by $\mathbf{z}_n$. The target is described by its state $\mathbf{x}_t$, with known a priori distribution $p(\mathbf{x}_0)$. The target moves in discrete time slots[1] according to the following mobility model

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_t), \tag{1}$$

where $f(\cdot)$ is a mobility function, and $\mathbf{u}_t$ is process noise. At time t, a subset $\mathcal{M}_t$ of sensors can measure some physical quantity related to the state of the object, e.g., range, speed, or angle. Here, we will consider $\mathcal{M}_t$ as the sensors within sensing range r from the target. We denote the measurement of sensor n at time t by

$$y_{n,t} = g_n(\mathbf{x}_t, \mathbf{v}_{n,t}), \tag{2}$$

[1]We assume sufficient synchronization among the nodes to enable time division scheduling (see Section IV).

where $\mathbf{v}_{n,t}$ is observation noise, assumed to temporally white and independent from sensor to sensor. We have omitted the dependence of $g_n(\cdot)$ on the known sensor position $\mathbf{z}_n$ for notational convenience. We further introduce $\mathbf{y}_t = \{y_{n,t}\}_{n\in\mathcal{M}_t}$. The goal of the sensor network is for every sensor to determine $b(\mathbf{x}_t) \doteq p(\mathbf{x}_t|\mathbf{y}_{1:t})$.

B. Communication Model

Sensors can communicate with neighboring sensors within communication range r [14], using packets containing N_{data} bits, sent with a rate R_{data} bits/s. Communication is over a common channel and is coordinated through a time-division multiple access (TDMA) protocol. All transmissions that do not collide are assumed to be error-free.

III. TARGET TRACKING

We use a Bayesian filtering approach, where we recursively determine

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int b(\mathbf{x}_{t-1})p(\mathbf{x}_t|\mathbf{x}_{t-1})\mathrm{d}\mathbf{x}_{t-1}, \tag{3}$$

in which $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ is obtained from (1), and

$$\begin{aligned} b(\mathbf{x}_t) &\propto p(\mathbf{x}_t|\mathbf{y}_{1:t-1})p(\mathbf{y}_t|\mathbf{x}_t) & (4)\\ &= p(\mathbf{x}_t|\mathbf{y}_{1:t-1})\prod_{n\in\mathcal{M}_t} p(y_{n,t}|\mathbf{x}_t), & (5)\end{aligned}$$

where the last transition is due to the independence of the measurements, and where $p(y_{n,t}|\mathbf{x}_t)$ is obtained from (2). We initialize (5) with $p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = p(\mathbf{x}_0)$ for $t = 0$. To enable efficient computation of (3)–(5), we resort to a particle filter solution [15].

A. Centralized Tracking

We first describe a centralized particle filter (CPF), in which at every time slot, a fusion center (FC) collects all observations from the sensors to determine $b(\mathbf{x}_t)$. At the end of time slot $t-1$, the fusion center has a particle representation of $b(\mathbf{x}_{t-1})$, given by $\{\mathbf{x}_{t-1}^{(k)}, w_{t-1}^{(k)}\}_{k=1}^{N_p}$. To compute a particle representation of $b(\mathbf{x}_t)$, the FC proceeds as follows: (i) for every particle $\mathbf{x}_{t-1}^{(k)}$, we generate a new particle $\mathbf{x}_t^{(k)}$ from a proposal distribution $q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(k)})$; (ii) we adjust the weight of the k-th particle as

$$w_t^{(k)} \propto w_{t-1}^{(k)}\frac{p(\mathbf{y}_t|\mathbf{x}_t^{(k)})p(\mathbf{x}_t^{(k)}|\mathbf{x}_{t-1}^{(k)})}{q(\mathbf{x}_t^{(k)}|\mathbf{x}_{t-1}^{(k)})}, \tag{6}$$

assuming the weights are normalized; (iii) the estimate of $\mathbf{x}_t$ is given by

$$\hat{\mathbf{x}}_t = \sum_{k=1}^{N_p} w_t^{(k)}\mathbf{x}_t^{(k)}. \tag{7}$$

(iv) to avoid degeneracy problems, resampling is commonly applied, resulting in N_p equal-weight samples $\{\mathbf{x}_t^{(k)}, 1/N_p\}_{k=1}^{N_p}$, which form a particle representation of $b(\mathbf{x}_t)$.

While in general $q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(k)})$ can also depend on $\mathbf{y}_t$, a popular choice for $q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(k)})$ is $p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(k)})$, so that the particles $\{\mathbf{x}_t^{(k)}, w_{t-1}^{(k)}\}_{k=1}^{N_p}$ can be interpreted as a particle representation of $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$, and (6) reverts to $w_t^{(k)} \propto w_t^{(k-1)}p(\mathbf{y}_t|\mathbf{x}_t^{(k)})$. It should be noted that (6) can be further expanded as (assuming sampling from $p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(k)})$)

$$w_t^{(k)} \propto w_{t-1}^{(k)} \prod_{n\in\mathcal{M}_t} p(y_{n,t}|\mathbf{x}_t^{(k)}). \tag{8}$$

B. Distributed Tracking using Belief Consensus

As the use of a fusion center is not scalable for large WSN, we consider a DPF [1]. Assuming each sensor has access to the same particle representation $\{\mathbf{x}_{t-1}^{(k)}, w_{t-1}^{(k)}\}_{k=1}^{N_p}$, and uses the same proposal density $q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(k)})$, the distributed particle filter (DPF) with standard belief consensus (SBC) is able to recursively compute $\{\mathbf{x}_t^{(k)}, w_t^{(k)}\}_{k=1}^{N_p}$ in a fully decentralized manner, such that every node has the same particle representation of $b(\mathbf{x}_t)$. DFP-SBC operates as follows:

1) *Sampling:* Every sensor generates the *same* samples $\mathbf{x}_t^{(k)}$ from $q(\mathbf{x}_t|\mathbf{x}_{t-1}^{(k)})$. This can be achieved by setting the seed state in the pseudorandom number generators of the sensors equally.
2) *Local weight computation:* Every sensor computes a local weight

$$w_{n,t}^{(k)} = \begin{cases} p(y_{n,t}|\mathbf{x}_t^{(k)}) & n\in\mathcal{M}_t\\ 1 & \text{else.}\end{cases} \tag{9}$$

3) *Average weight consensus:* The sensors run a two-stage distributed consensus algorithm to compute $\prod_n w_{n,t}^{(k)} = \prod_{n\in\mathcal{M}_t} p(y_{n,t}|\mathbf{x}_t^{(k)})$, which is used to update the common weights $w_t^{(k)}$ in (6). The nodes first run average consensus on $\log w_{n,t}^{(k)}$:

$$\begin{aligned}\log w_{n,t}^{(k,i)} &= \log w_{n,t}^{(k,i-1)} & (10)\\ &+ \xi(r)\sum_{u\in\mathcal{N}_n}\left(\log w_{u,t}^{(k,i-1)} - \log w_{n,t}^{(k,i-1)}\right),\end{aligned}$$

where $\xi(r)$ is generally chosen to be smaller than the inverse of the maximum node degree[2] $d_{\max}(r)$, $\mathcal{N}_n$ is the set of sensors in the neighborhood of sensor n, and i is an iteration index, with $\log w_{n,t}^{(k,0)} = \log w_{n,t}^{(k)}$. It can be shown that as $i\to+\infty$, $\log w_{n,t}^{(k,i)} \to \sum_n \log w_{n,t}^{(k)}/N_s$. Note that (10) involves every node broadcasting $\left\{\log w_{n,t}^{(k,i-1)}\right\}_{n=1}^{N_p}$ to all of its neighbors, receiving $\log w_{u,t}^{(k,i-1)}$ from all of its neighbors, and then computing (10).
4) *Max Weight consensus:* After a pre-defined number of iterations, say I, (10) is halted and max-consensus is run to achieve *exact* agreement on the weights

$$\log w_{n,t}^{(k,I+j)} = \max_{u\in\mathcal{N}_n}\left(\log w_{n,t}^{(k,I+j-1)}, \log w_{u,t}^{(k,I+j-1)}\right), \tag{11}$$

[2]To be more concrete, $\xi(r) \in [0, 1/\lambda_{\max}]$, where $\lambda_{\max}$ is the largest eigenvalue of the graph Laplacian. We chose $\xi(r) = 1/d_{\max}(r)$, when $d_{\max}(r) \geq \lambda_{\max}$ and $\xi(r) = 1/(d_{\max}(r)+1)$, when $d_{\max}(r) < \lambda_{\max}$.

for $j = 0, 1, 2, \ldots$ Max-consensus is also implemented by broadcasts, and converges to $\log \tilde{w}_t^{(k)} = \max_n \log w_{n,t}^{(k,I)} \approx \sum_n \log w_{n,t}^{(k)}/N_s$ after D_r iterations, where D_r is the graph diameter.

It should be noted that other exact consensus algorithms can be executed instead of max-consensus, for example min-consensus. How well $\log \tilde{w}_t^{(k)}$ approximates the desired value $\sum_n \log w_{n,t}^{(k)}/N_s$ depends on I and the structure of the communication graph. Multiplying $\log \tilde{w}_t^{(k)}$ by N_s, taking exponentials, and normalizing, gives us a fully distributed way to compute (6).

IV. Communication Delay

In this section, we will determine the delay incurred when performing I iterations of average consensus, followed by D_r iterations of max-consensus, based on the communication model from Section II-B. As both algorithms are based on broadcasts, we first determine the time required for every node to perform a broadcast. The time will be measured in micro-slots. Note that an iteration takes multiple micro-slots (depending on the schedule, and on the size of the packets), and that in a time-slot $I + D_r$ iterations are executed.

A. Time per Iteration

We represent the sensor network by a communication graph $G = (V, E)$ consisting of N_s vertices (nodes) and E edges, where $(n, n') \in E$, when n and n' are within a distance of r. We aim to find the minimum number of time slots required to perform one broadcast for all the nodes in the network, based on spatial time division multiple access (STDMA). In graph theory, this is called node or vertex coloring. Vertex coloring can be formulated mathematically as an integer programming problem [16] as follows, assuming to have a maximum of $T = N_s$ slots available:

$$\begin{aligned}
\text{minimize} \quad & \sum_{\tau=1}^{T} s_\tau && && (12)\\
\text{s.t.} \quad & \sum_{n=1}^{N_s} s_{n\tau} \le s_\tau N_s, && \forall \tau \in T && (13)\\
& \sum_{\tau=1}^{T} s_{n\tau} = 1, && \forall n \in V && (14)\\
& s_{n\tau} + \sum_{k \in \mathcal{N}_n} s_{k\tau} \le 1, && \forall n \in V, \tau \in T && (15)\\
& s_\tau \in \{0,1\}, && \forall \tau \in T && (16)\\
& s_{n\tau} \in \{0,1\}, && \forall n \in V, \tau \in T && (17)
\end{aligned}$$

where $s_\tau = 1$ when a transmission occurs in micro-slot τ, $s_{n\tau} = 1$ is one when node n broadcasts in slot τ. The objective (12) aims to minimize the total number of micro-slots. The constraints impose the following: (13) specifies that the number of nodes scheduled in one slot should not be more than the total number of nodes in the network; (14) states that every node must be assigned a slot in the schedule; (15) ensures that for successful transmission of a packet, a node should not transmit and receive at the same time; and, finally, constraints (16) and (17) imposes the integer requirements on the optimization variables. The above formulation is binary integer programming (BIP), which we solved to optimality using `CVX` [17] with support of MILP programming from `MOSEK` [18]. The solution in terms of the number of micro-slots is denoted by N_r^*. While our scheduler is centralized, we note that for a distributed schedule, the number of micro-slots will be at least N_r^*.

Rather than solving the optimization problem exactly, we can also use $d_{\max}(r) + 1 \le N_r^*$ as a lower bound.

B. Total Time

The average consensus requires I iterations, and max-consensus requires an additional D_r iterations, which varies with r. In every micro-slot, we can send packets of N_{data} bits, at a rate of R_{data} bits per second, thus taking $N_{\text{data}}/R_{\text{data}}$ seconds. Assuming the weights are quantized with q bits, nodes broadcast qN_p bits per iteration, requiring $\lceil qN_p/N_{\text{data}} \rceil$ packets, requiring $\lceil qN_p/N_{\text{data}} \rceil N_{\text{data}}/R_{\text{data}}$ seconds. Hence, the total duration to run DPF per time slot takes

$$T_{\text{DPF}}\,[\text{s}] = N_{\text{tot},r} \lceil qN_p/N_{\text{data}} \rceil N_{\text{data}}/R_{\text{data}}, \tag{18}$$

where we have introduced the total number of micro-slots per time slot T_s as

$$N_{\text{tot},r} \doteq N_r^* \left(I + D_r\right). \tag{19}$$

Note that for tracking to operate, $T_{\text{DPF}} \le T_s$, where T_s is the time slot duration.

C. Relation Between Time and Performance

We consider a fixed number of nodes, N_s. The error in average consensus goes down exponentially with $1 - \xi(r)\lambda_2(r)$, where $\lambda_2(r)$ is the spectral gap of the graph Laplacian. Hence, after I iterations, the relative disagreement [3] $0 < \delta < 1$ is [19]

$$\delta \approx \left(1 - \xi(r)\lambda_2(r)\right)^I. \tag{20}$$

In other words, to achieve a relative disagreement of δ, we can solve (20) for I, substitute in (19), and find that in total we require

$$N_{\text{tot},r} = \left(\left\lceil \frac{\log \delta}{\log\left(1 - \xi(r)\lambda_2(r)\right)} \right\rceil + D_r \right) N_r^* \tag{21}$$

micro-slots for average consensus and max-consensus. For large r, $\lambda_2(r) \to N_s$, $\xi(r) \to 1/N_s$, $N_r^* \to N_s$, and $D_r \to 1$, so that the number of micro-slots will tend to N_s, irrespective of δ. For $r \to r_{\min}$, the smallest radius for a connected network, $\lambda_2(r) \to 0$, so that the number of iterations is essentially $\lceil \log \delta \rceil$, and the total time becomes

[3]The relative disagreement is defined as

$$\delta = \left\| \log \mathbf{w}_t^{(k,i)} - \frac{\mathbf{1}}{N} \sum_n \log w_{n,t}^{(k,0)} \right\| \Big/ \left\| \log \mathbf{w}_t^{(k,0)} - \frac{\mathbf{1}}{N} \sum_n \log w_{n,t}^{(k,0)} \right\|,$$

where $\log \mathbf{w}_t^{(k,i)} = [\log w_{1,t}^{(k,i)}, \ldots, \log w_{N,t}^{(k,i)}]^T$. Note that though δ depends on the specific time t and the specific particle k, we simply write δ for notational convenience.

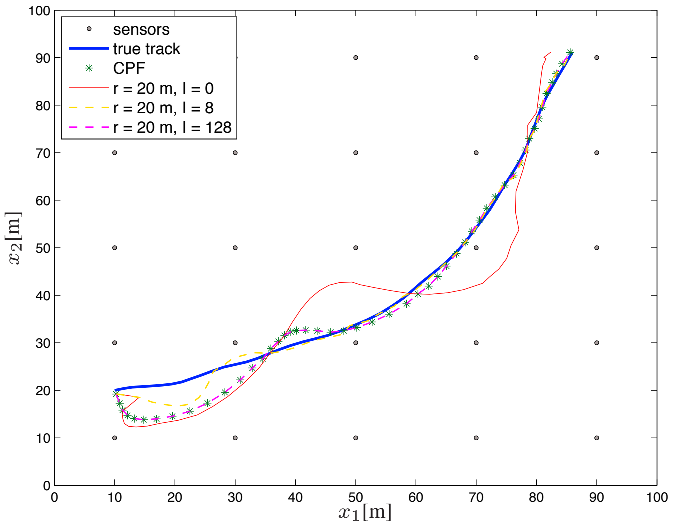

Fig. 1. The target movement through the area surveyed by the network, the CPF estimate and 3 estimates using DPF-SBC with same communications radius, but a different number of SBC iterations I. Clearly the DPF-SBC estimates get better with respect to CPF as I increases.

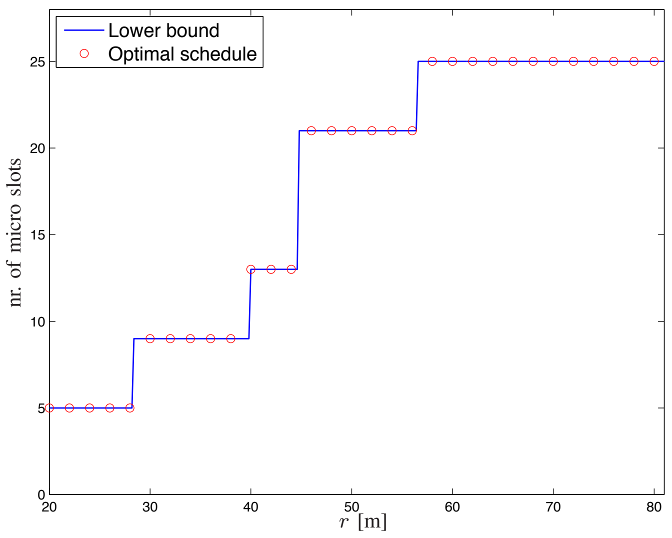

Fig. 2. The solid line shows the lower bound of micro slots for scheduling a 5×5 square network with 25 sensors placed 20 m apart. The rings shows the achieved optimal schedule solving the optimization problem from Section IV-A.

$N_{\text{tot},r} = (\lceil \log \delta \rceil + D_r) N^*_{r_{\min}}$ micro-slots. Hence, for large allowable disagreement, only when $D_r N^*_{r_{\min}} < N_s$, it is advantageous to have a small r. For very small allowable disagreement, it is preferred to have large r.

V. Numerical Results

A. Simulation Setup

We consider a network of 25 sensors placed in a square grid with a minimum distance of 20 m between two neighboring sensors (see Fig. 1). The target's state is given by its two-dimensional position $[x_{1,t}, x_{2,t}]$ and its two-dimensional velocity $[\dot{x}_{1,t}, \dot{x}_{2,t}]$, so that $\mathbf{x}_t = [x_{1,t}\ \ x_{2,t}\ \ \dot{x}_{1,t}\ \ \dot{x}_{2,t}]^T$. At time $t = 0$, the target state has a known circular Gaussian a priori distribution $p(\mathbf{x}_0)$, with standard deviation 2 meters and a properly chosen mean. The target moves through the area surveyed by the sensors according to a white Gaussian acceleration model, described by

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u}_t, \tag{22}$$

where

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & T_s & 0 \\ 0 & 1 & 0 & T_s \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} T_s^2/2 & 0 \\ 0 & T_s^2/2 \\ T_s & 0 \\ 0 & T_s \end{bmatrix} \tag{23}$$

where $\mathbf{u}_t = [u_{1,t}\ u_{2,t}]^T$ and $u_{1,t}, u_{2,t} \sim \mathcal{N}(0, 0.04)$, the sampling interval $T_s = 1$ second. The target moves for a duration of 50 seconds. The sensing range is set to $r_{\text{sense}} = +\infty$, so that all sensors collect range measurements every second

$$y_{n,t} = \|\mathbf{z}_n - (x_{1,t}, x_{2,t})\| + v_{n,t}, \tag{24}$$

where $v_{n,t} \sim \mathcal{N}(0, 1\,\text{m}^2)$. The communication range r is varied between the minimum and maximum distance separating two sensors (i.e., $r \in [20\,\text{m}, 114\,\text{m}]$). The SBC algorithm is evaluated for a fixed number of particles ($N_p = 256$) a varying number of iterations $I \in \{0, 1, 2, 4, \ldots, 256\}$. Note that $I = 0$ means that we only run max-consensus. In terms of communication capabilities, we assume a packet size is set to $N_{\text{data}} = 256$ kB, a 2^8-level quantization (i.e., $q = 8$) of the particle weights, and a data rate of $R_{\text{data}} = 500\,\text{kbit/s}$ leading to micro-slots of 4.1 ms.

As performance measure, we consider the degradation with respect to the CPF, running with the same number of particles. Let $\hat{\mathbf{x}}_{t,\text{C}}$ be the CPF estimate, and $\hat{\mathbf{x}}_{t,\text{D}}$ the DPF-SBC estimate (which depends on I and r), then the relative error is defined as

$$e_{\text{rel}} = \mathbb{E}\left\{ \frac{\|\hat{\mathbf{x}}_{t,\text{C}} - \hat{\mathbf{x}}_{t,\text{D}}\|}{\|\hat{\mathbf{x}}_{t,\text{C}} - \mathbf{x}_t\|} \right\}, \tag{25}$$

where the expectation is taken over a steady state time window (here set between $t \in [10\,\text{s}, 40\,\text{s}]$). The relative error e_{rel} is estimated through 200 Monte Carlo runs, where the target track is kept constant, but new observations are generated in each trial. The target track is visualized in Fig. 1. The total time to run DPF-SBC per time slot is computed as stated in (18), and the time to take the measurement is disregarded.

B. Simulation Results and Discussion

We first show some indicative performance results for $r = 20$ m, for varying number of iterations $I \in \{0, 8, 128\}$ in Fig. 1. We see that the CPF closely follows the true track, and that DPF-SBC is able to achieve good performance for $I \geq 8$. When only max-consensus is performed (i.e., $I = 0$), large tracking errors are visible.

Fig. 2 shows the number of micro-slots required to perform one iteration in SBC, as a function of r. Both the outcome from the optimization (12) as well as the lower bound $d_{\max}(r) + 1$

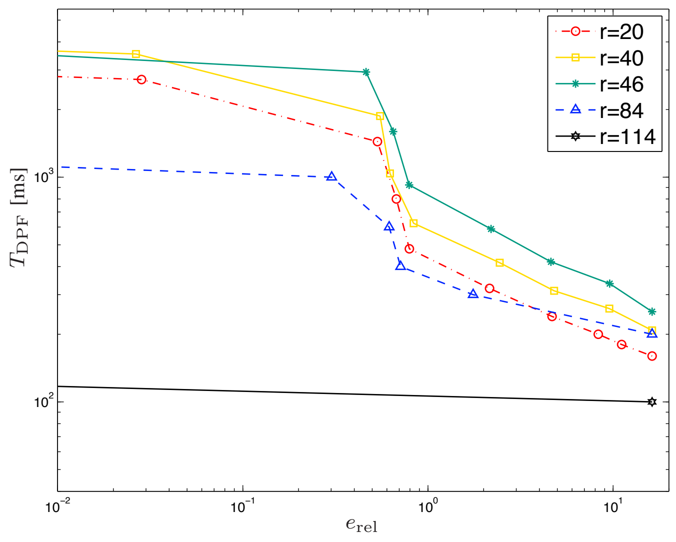

Fig. 3. The delay time in milliseconds plotted against the error for 5 different communication ranges represented by the 5 curves, when performing exact consensus by max-consensus. As we increase I the error decreases, but the delay time increases.

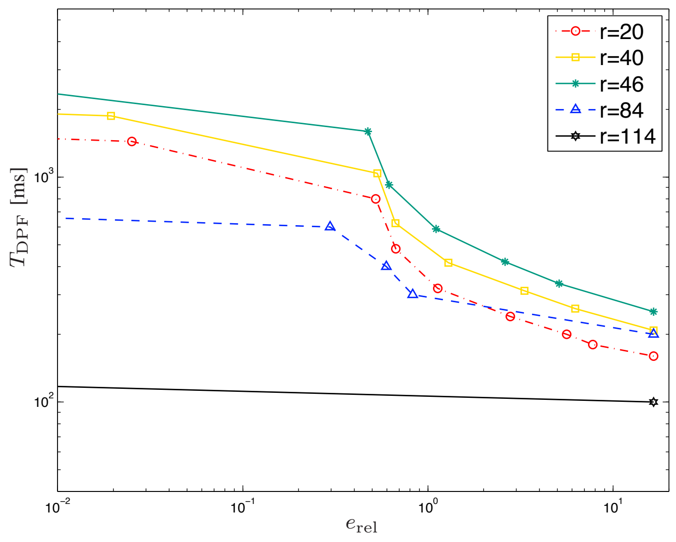

Fig. 4. The same simulations as in Fig. 3, but where min-consensus is run instead of max-consensus to achieve exact consensus on the weights.

are shown. We see that the number of slots varies from 5 to 25 and that the lower bound is tight.

Fig. 3 shows the total time $T_{\rm DPF}$ for different r, and a varying number of iterations, as a function of the achieved $e_{\rm rel}$ when running max-consensus in the final stage of the algorithm. When $I = 0$, $e_{\rm rel}$ is independent of r but the delay $T_{\rm DPF}$ increases when r is reduced, as a reduction in r results in a larger graph diameter. For a fixed r, increasing I decreases the error exponentially. However, for a fixed I, increasing r does not always impact the error greatly. For example, increasing r from 40 m to 46 m barely changes the error but results in significant additional scheduling delays. For this particular network, setting $r = 46$ m gives the worst performance/delay trade-off. Selecting a communication range that covers the whole network gives the best performance/delay trade-off. An interesting thing to note is the slow increase of the delay for small r. This highlights the dominating effect of the max-consensus in the delay, when the number of SBC iterations is small. The max-consensus delay affects the smaller communication ranges to a much larger extent due to the slower convergence of the max-consensus in those networks, which is proportional to the communication diameter of the network. This is an important practical consideration.

Finally, in Fig. 4, we show the performance/delay trade-off when running min-consensus to achieve exact consensus on the weights. We observe that min-consensus gives a gain in performance over max-consensus. The difference in performance is due to min-consensus being more similar to the operation of multiplication of the weights than max-consensus. Hence, min-consensus approximates the CPF better, which leads to a smaller error for the considered scenario.

VI. Conclusions

We have studied the impact of TDMA scheduling delays in distributed target tracking. Our focus was on the DPF-SBC, which is known to benefit from highly connected networks, but may suffer large scheduling delays, due to increased interference. We found that for small regular networks, limited communication ranges are not recommended, despite fast schedules: they lead to slow convergence of the SBC, and also require a relatively long delays for max-consensus. The best performance/delay trade-off is achieved when the communication range is set to cover the whole network when the network is small. For practical systems, where only few nodes will be involved in DPF-SBC, our findings allow for very simple scheduling, as only one sensor at the time can be allowed to broadcast. For very large networks, highly irregular networks, or networks with limited sensing range, this conclusion may no longer hold, and warrants further investigation.

Acknowledgment

This research was supported in part, by the European Research Council, under Grant No. 258418 (COOPNET), the Swedish Research Council, under Grant No. 2010-5889, the Swedish Foundation for Strategic Research (SSF), and ELLIIT.

References

[1] O. Hlinka, F. Hlawatsch, and P. Djuric, “Distributed particle filtering in agent networks: A survey, classification, and comparison,” *IEEE Signal Processing Magazine*, vol. 30, no. 1, pp. 61–81, 2013.

[2] R. Olfati-Saber, “Kalman-consensus filter: Optimality, stability, and performance,” in *IEEE Conference on Decision and Control*, pp. 7036–7042, 2009.

[3] S. Farahmand, S. Roumeliotis, and G. Giannakis, “Set-membership constrained particle filter: Distributed adaptation for sensor networks,” *IEEE Transactions on Signal Processing*, vol. 59, no. 9, pp. 4122–4138, 2011.

[4] F. Meyer, E. Riegler, O. Hlinka, and F. Hlawatsch, “Simultaneous distributed sensor self-localization and target tracking using belief propagation and likelihood consensus,” *arXiv preprint arXiv:1211.6988*, 2012.

[5] D. Gu, "Distributed em algorithm for gaussian mixtures in sensor networks," *IEEE Transactions on Neural Networks*, vol. 19, no. 7, pp. 1154–1166, 2008.

[6] A. Mohammadi and A. Asif, "Consensus-based distributed unscented particle filter," in *IEEE Statistical Signal Processing Workshop (SSP)*, pp. 237–240, 2011.

[7] R. Olfati-Saber, E. Franco, E. Frazzoli, and J. Shamma, "Belief consensus and distributed hypothesis testing in sensor networks," *Networked Embedded Sensing and Control*, pp. 169–182, 2006.

[8] O. Hlinka, O. Sluciak, F. Hlawatsch, P. Djuric, and M. Rupp, "Distributed gaussian particle filtering using likelihood consensus," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3756–3759, 2011.

[9] A. Kashyap, T. Başar, and R. Srikant, "Quantized consensus," *Automatica*, vol. 43, no. 7, pp. 1192–1203, 2007.

[10] A. Nedic, A. Olshevsky, A. Ozdaglar, and J. Tsitsiklis, "On distributed averaging algorithms and quantization effects," *IEEE Transactions on Automatic Control*, vol. 54, no. 11, pp. 2506–2517, 2009.

[11] T. Aysal, M. Coates, and M. Rabbat, "Distributed average consensus with dithered quantization," *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 4905–4918, 2008.

[12] V. Saligrama, M. Alanyali, and O. Savas, "Distributed detection in sensor networks with packet losses and finite capacity links," *Signal Processing, IEEE Transactions on*, vol. 54, no. 11, pp. 4118–4132, 2006.

[13] S. Vanka, M. Haenggi, and V. Gupta, "Convergence speed of the consensus algorithm with interference and sparse long-range connectivity," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 855–865, 2011.

[14] B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit disk graphs," *Discrete Mathematics*, vol. 86, no. 1-3, pp. 165–177, 1990.

[15] M.S. Arulampalam, S. Maskell, N. Gordon and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, pp. 174–188, 2002.

[16] P. Värbrand, D. Yuan, and P. Björklund, "Resource optimization of spatial tdma in ad hoc radio networks: A column generation approach," *INFOCOM*, 2003.

[17] CVX Research, Inc., "CVX: Matlab software for disciplined convex programming, version 2.0 beta," Sept. 2012.

[18] The MOSEK optimization software, http://www.mosek.com/.

[19] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.