Distributed Kalman Filtering Using Weighted Averaging

Alriksson, Peter; Rantzer, Anders

2006

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

Distributed Kalman Filtering Using Weighted Averaging

Peter Alriksson[1] and Anders Rantzer[1]

Abstract— **This paper addresses the problem of distributed Kalman filtering, with focus on limiting the required communication bandwidth. By distributed we refer to a scenario when all nodes in the network desire an estimate of the full state of the observed system and there is no centralized computation center. Communication only takes place between neighbors and only once each sample. To reduce bandwidth requirements of individual nodes, estimates instead of measurements are communicated. A new estimate is then formed as a weighted average of the neighbouring estimates. The weights are optimized to yield a small estimation error covariance in stationarity. The minimization can be done off line thus allowing only estimates to be communicated. The advantage of communicating estimates instead of measurements becomes more evident when the number of nodes exceeds the size of the state vector to be estimated. The algorithm is applied to one simple second order system and temperature sensing network.**

I. Introduction

As battery and processing power of nodes in sensor networks increases the possibility of more intelligent estimation schemes become more and more important. The use of sensor networks was first driven by military applications, but with cheaper technology many other areas could make use of sensor networks, see for example [1] and [2]. Advantages with wireless sensor networks typically include more robustness, because more than one unit is performing the same task and increased flexibility.

However decentralized estimation is a far more complex task than traditional centralized estimation. At least two big questions arise, what to send and how to make use of the received information. The first question has two obvious candidates, measurements or estimates.

In the case when measurements are transmitted the main problem is that all nodes usually do not have the possibility to communicate with all other nodes, thus the measurements have to be routed through other nodes. These nodes might experience a very high communication load, thus limiting the size of the network. This is especially evident when the number of nodes exceed the size of the state vector to be estimated. Once the routing problem is solved, generating an estimate based on measurements is straight forward, at least if the system is linear and subject to Gaussian disturbances.

The case where estimates are communicated has been given great attention in the literature. In for example [3] a decentralized Kalman filter was proposed. However, this algorithm requires every node to be able to communicate with every other node, which might not be possible. An alternative approach is to only allow nodes to communicate with their neighbors. As opposed to the case where measurements are communicated no routing is required when estimates are used as information carriers.

Without direct communication between all nodes a new problem is introduced, namely how to combine estimates from just neighboring nodes. To optimally combine two estimates one has to know the mutual information between the estimates. Computing this quantity for a general communication graph is a difficult task that requires global knowledge of the topology. In the case of a loop-free graph the problem was solved in [4] by introduction of a channel filter. This approach was used in a coordinated search strategy application, see [5].

If the state is assumed constant, the problem can be viewed as a distributed average problem which has been studied by for example [6]. The problem was generalized to time varying states in [7] and [8] using consensus filters.

A closely related area is how to combine estimates from a number of non-communicating estimators, so called track-to-track fusion algorithms. This problem differs in that the combined estimate is not used to compute a new estimate in the individual nodes. The fused estimate is only communicated to a central node to be used for some task. This problem was studied in for example [9] and [10].

II. Problem Formulation

Consider the following discrete-time linear system

$$x(k+1) = Ax(k) + v(k) \tag{1}$$

where $x(k) \in \mathbf{R}^n$ is the state of the system and $v(k) \in \mathbf{R}^n$ is a stochastic disturbance. The disturbance is assumed to be a white zero mean Gaussian process with covariance defined below.

The process is observed by N agents each with some processing and communication capability. The agents are labeled $i = 1, 2, \dots, N$ and form the set V. The communication topology is modeled as a graph $G = (V, E)$, where the edge (i, j) is in E if and only if node i and node j can exchange messages. The nodes to which a node communicates are called neighbors and are contained in the set N_i. Note that node i is also included in the set N_i.

Each node observes the process (1) by a measurement $y_i(k) \in \mathbf{R}^{p_i}$ of the following form

$$y_i(k) = C_i x(k) + e_i(k) \tag{2}$$

where $e_i(k) \in \mathbf{R}^{p_i}$ is a white zero mean Gaussian process. The measurement- and process disturbances are correlated according to

$$E \begin{bmatrix} v(k) \\ e_1(k) \\ \vdots \\ e_N(k) \end{bmatrix} \begin{bmatrix} v(l) \\ e_1(l) \\ \vdots \\ e_N(l) \end{bmatrix}^T = \begin{bmatrix} R_v & 0 & \dots & 0 \\ 0 & R_{e11} & \dots & R_{e1N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & R_{eN1} & \dots & R_{eNN} \end{bmatrix} \delta_{kl} \tag{3}$$

where $\delta_{kl} = 1$ only if $k = l$. Note that this is a heterogeneous setup where each agent is allowed to to take measurements of arbitrary size and precision. Further the disturbances acting on the measurements are allowed to be correlated.

Each node is only allowed to communicate with its neighbors and only once between each measurement. Further the only assumption made on the graph structure is that it has to be connected, other assumptions such as requiring it to be loop free is not necessary. No node is superior to any other and thus no central processing is allowed. This setup is somewhat different from the setup used in for example distributed control problems where each node in the graph also has dynamics associated with it. The reader should think of the problem studied here as for example a network of sensors trying to estimate the position of an external object they observe.

The goal is to make sure the every node in the network has a good estimate $\hat{x}_i(n)$ of the state $x(n)$.

III. Sharing Measurements

Taking the Bayesian point of view, the best possible estimate of the state at time k can be computed from the conditional probability distribution of x given all past measurements up to time k.

In the case of the communication restrictions imposed above, this has to be restricted to all available past measurements. Which past measurements that are available depends on the diameter of the graph, that is the maximum distance between two nodes. One way to provide all nodes with all measurements would be to tag all measurements with time and origin and then flood the network. However, as the number of nodes grows, the bandwidth requirement will make this approach *practically unattractive*.

Even if the optimal scheme is practically unattractive it still serves as a good comparison to other approaches. Therefore we will show one way to compute the estimation error covariance for this case using a simple example.

Now let $Y_i(k)$ denote the information available to node i at time k. For example, node 1 in in Figure 1 has access to $Y_1(k) = \{y_1(k), y_2(k), y_3(k-1), y_4(k-2)\}$, whereas $Y_3(k) = \{y_1(k-1), y_2(k), y_3(k), y_4(k)\}$ (note that $Y_i(k-1), Y_i(k-2)$ and so on has already been incorporated in the estimate as a normal Kalman filter will be used). If the state of the system is augmented with delayed versions of the original state, the new system in node 3 for example becomes

$$\begin{bmatrix} x(k+1) \\ x(k) \end{bmatrix} = \begin{bmatrix} A & 0 \\ I & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ x(k-1) \end{bmatrix} + \begin{bmatrix} I \\ 0 \end{bmatrix} v(k). \tag{4}$$

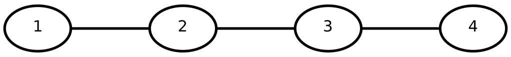

Fig. 1. Simple graph.

The corresponding augmented measurement equation is

$$\begin{bmatrix} y_1(k-1) \\ y_2(k) \\ y_3(k) \\ y_4(k) \end{bmatrix} = \begin{bmatrix} 0 & C_1 \\ C_2 & 0 \\ C_3 & 0 \\ C_4 & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ x(k-1) \end{bmatrix} + \begin{bmatrix} e_1(k-1) \\ e_2(k) \\ e_3(k) \\ e_4(k) \end{bmatrix}. \tag{5}$$

If the augmented system is detectable the optimal estimator is the normal Kalman filter, and the optimal estimation error covariance is the solution to the associated Riccati equation.

IV. Sharing Estimates

The approach described above requires every node in the graph to send its measurements to every other node. For a graph of moderate size this might be feasible, however as the size of the graph increases some nodes might experience very high communication load. If the dimension of the state vector is small compared to the number of nodes in the network, a much more bandwidth saving approach is to share estimates with the neighbouring nodes.

Having removed the potential congestion problem, a new difficulty is introduced, namely how to combine estimates in an optimal way. The problem is that estimates are not independent, as they contain the same process noise, and possibly also the same measurement information. To optimally combine two estimates the mutual information must be subtracted.

In the case of a graph without loops the problem was solved in [4] by the use of an on-line channel filter. The channel filter basically keeps track of the mutual information between the nodes by looking at what they send to each other and thus allows for the estimates to be merged in an optimal way.

For a graph with loops, two nodes can not compute the mutual information by just using local information. Information can for example travel from node A to node C and then to node B. When node A and B are to compute their mutual information they do not know about the information that was sent through C.

To solve the problem for a general communication topology a global off-line method will be used. Instead of subtracting the mutual information, the estimates are weighed so that the covariance of the merged estimate is minimized. This approach will not give the optimal solution, but is is applicable to graphs with loops. Weighted averaging can be seen as a generalization of the two-sensor track-fusion algorithm presented in [11]. To evaluate performance the resulting estimation error covariance can easily be compared to the optimal as computed in section III.

A. On-line computations

The algorithm consists of the two traditional estimation steps, measurement update and prediction together with an additional step where the nodes communicate and merge estimates. We will refer to an estimate after measurement update as local and after the communication step as regional.

1) **Measurement update**
   The local estimate $\hat{x}_i^{local}(k|k)$ is formed by the predicted regional estimate $\hat{x}_i^{reg}(k|k-1)$ and the local measurement $y_i(k)$

   $$\hat{x}_i^{local}(k|k) = \hat{x}_i^{reg}(k|k-1) + K_i[y_i(k) - C_i\hat{x}_i^{reg}(k|k-1)]. \tag{6}$$

   where K_i is computed off-line. The predicted estimate at time zero is defined as $\hat{x}_i^{reg}(0|-1) = \hat{x}_0$ where $\hat{x}_0$ is the initial estimate of $x(0)$.
2) **Merging**
   First the agents exchange their estimates over the communication channel. This communication is assumed to be error and delay free. The merged estimate $\hat{x}_i^{reg}(k|k)$ in node i is defined as a linear combination of the estimates in the neighboring nodes N_i.

   $$\hat{x}_i^{reg}(k|k) = \sum_{j \in N_i} W_{ij}\hat{x}_j^{local}(k|k) \tag{7}$$

   The weighting matrices W_{ij} are computed off-line by the procedure described in section IV-B.
3) **Prediction**
   Because the measurement- and process noises are independent the prediction step only includes

   $$\hat{x}_i^{reg}(k+1|k) = A\hat{x}_i^{reg}(k|k) \tag{8}$$

B. Off-line computations

To be able to execute the steps described above the parameters $K_i \in \mathbf{R}^{n \times p_i}$ and $W_{ij} \in \mathbf{R}^{n \times n}$ must be chosen. In this section an iterative algorithm will be developed for this purpose. First let the estimation error in node i be defined as

$$\tilde{x}_i(k) = x(k) - \hat{x}_i(k) \tag{9}$$

with covariance

$$P_{ij}(k) = E\tilde{x}_i(k)\tilde{x}_j(k)^T. \tag{10}$$

Now note that the estimation error covariance (10) after step 1) above can be written as

$$P_{ij}^{local}(k|k) = (I - K_i(k)C_i)P_{ij}^{reg}(k|k-1)(I - K_j(k)C_j)^T + K_iR_{eij}K_j^T. \tag{11}$$

with $P_{ij}^{reg}(0|-1) = P_0$ where P_0 is the initial estimation error covariance. Minimizing P_{ii}^{local} with respect to $K_i(k)$ gives

$$K_i(k) = P_{ii}^{reg}(k|k-1)C_i^T(R_{eii} + C_iP_{ii}^{reg}(k|k-1)C_i^T)^{-1}. \tag{12}$$

Next the choice of W_{ij} will be addressed. To keep the estimates unbiased the following constraint is introduced.

$$\sum_{j \in N_i} W_{ij}(k) = I_{n \times n} \tag{13}$$

Using (13) we can write the merged estimation error as

$$\begin{aligned}\tilde{x}_i^{reg}(k|k) &= x(k) - \hat{x}^{reg}(k|k) \\ &= \sum_{j \in N_i} W_{ij}(k)x(k) - \sum_{j \in N_i} W_{ij}(k)\hat{x}_j^{local}(k|k) \\ &= \sum_{j \in N_i} W_{ij}(k)\tilde{x}_j^{local}(k|k).\end{aligned} \tag{14}$$

The estimation error covariance matrices can thus be computed as

$$P_{ij}^{reg}(k|k) = \sum_{l \in N_i} \sum_{m \in N_j} W_{il}(k)P_{lm}^{local}(k|k)W_{jm}(k)^T \tag{15}$$

Equation (15) can be written in matrix form as

$$P^{reg}(k|k) = W(k)P^{local}(k|k)W(k)^T \tag{16}$$

by requiring that

$$W_{ij}(k) = 0 \quad \text{if } (i,j) \notin E. \tag{17}$$

The goal of the weight selection is to minimize the estimation error covariance matrix in each node in steady state. That is to minimize $P_{ii}^{reg}(k|k)$ for all i given the constraints (13) and (17) as k approaches infinity.

To simplify notation the time indexes will be dropped in the following part. The optimization problem for each node can thus be posed as

$$\begin{array}{cc} \min_{W_{i\cdot}} & P_{ii}^{reg} \\ \text{subject to} & \text{(13) and (17)} \end{array} \tag{18}$$

where $W_{i\cdot} = \begin{bmatrix} W_{i1} & \dots & W_{iN} \end{bmatrix}$. This minimization problem will be solved in a number of steps next. As (18) is a minimization of a quadratic matrix expression it will be minimized by completing the squares. First write the estimation error covariance in node i as

$$P_{ii}^{reg} = \begin{bmatrix} W_{i1} & \dots & W_{iN} \end{bmatrix} P^{local} \begin{bmatrix} W_{i1}^T \\ \vdots \\ W_{iN}^T \end{bmatrix}. \tag{19}$$

Introducing the sparsity constraint (17) is equivalent to removing the rows and columns corresponding to the weights that are required to be zero. Thus (19) can be written as

$$P_{ii}^{reg} = \bar{W}\bar{P}\bar{W}^T. \tag{20}$$

where $\bar{W}$ is of size $n \times (nm_i)$ and $\bar{P}$ is of size $nm_i \times (nm_i)$. Here m_i denotes the cardinality of N_i, that is the number of neighbors of node i.

Next the constraint (13) is introduced and (20) can be written as

$$P_{ii}^{reg} = \begin{bmatrix} I_{n \times n} & \bar{W}_{12} & \dots & \bar{W}_{1m_i} \end{bmatrix} Q \begin{bmatrix} I_{n \times n} \\ \bar{W}_{12}^T \\ \vdots \\ \bar{W}_{1m_i}^T \end{bmatrix} \tag{21}$$

The matrix Q is defined as

$$Q = \bar{P} - R - S \tag{22}$$

where

$$R = \begin{bmatrix} 0 & \bar{P}_{11} & \dots & \bar{P}_{11} \\ \bar{P}_{11} & -\bar{P}_{11} & \dots & -\bar{P}_{11} \\ \vdots & \vdots & \ddots & \vdots \\ \bar{P}_{11} & -\bar{P}_{11} & \dots & -\bar{P}_{11} \end{bmatrix} \tag{23}$$

and

$$S = \begin{bmatrix} 0 & & \dots & 0 \\ 0 & \bar{P}_{12} + \bar{P}_{21} & \dots & \bar{P}_{1m_i} + \bar{P}_{21} \\ 0 & \vdots & \ddots & \vdots \\ 0 & \bar{P}_{12} + \bar{P}_{m_i1} & \dots & \bar{P}_{1m_i} + \bar{P}_{m_i1} \end{bmatrix} \tag{24}$$

In the above equations all blocks of the $\bar{P}$ and $\bar{W}$ matrices are of size $n \times n$. Writing (21) as

$$P_{ii}^{reg} = \begin{bmatrix} I_{n\times n} & \bar{W}_{i\cdot} \end{bmatrix} \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \begin{bmatrix} I_{n\times n} \\ \bar{W}_{i\cdot}^T \end{bmatrix} \tag{25}$$

and completing the squares it can be seen that a $\bar{W}_{i\cdot}$ satisfying

$$\bar{W}_{i\cdot} Q_{22} = -Q_{12} \tag{26}$$

minimizes P_{ii}^{reg}. The matrix Q_{22} is in general not positive definite but positive semidefinite. This corresponds to that the solution of (26), if it exists, is not unique. To avoid that the magnitude of any weight becomes too large trace $\bar{W}\bar{W}^T$ is minimized subject to (26).

Next note that the estimation error covariance after the prediction step can be written as

$$P_{ij}^{reg}(k+1|k) = AP_{ij}^{reg}(k|k)A^T + R_v \tag{27}$$

Now all pieces are there to formulate an iterative algorithm for computing the stationary values of K_i and W_{ij}.

1) **Measurement update**
   Update the covariance matrices P_{ij}^{local} according to (11) and (12).
2) **Merging**
   Solve the N minimization problems (18).
   Update the covariance matrices P^{reg} according to (16).
3) **Prediction**
   Update the covariance P^{reg} according to (27).

The three steps; measurement update, merging and prediction are then iterated until a steady state value of the estimation error covariance matrix is achieved.

V. EXAMPLES

A. Simple Second Order System

Next a simple example with 11 nodes estimating the state of a second order system will be studied. The dynamics are given by

$$x(k+1) = \begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix} x(k) + v(k). \tag{28}$$

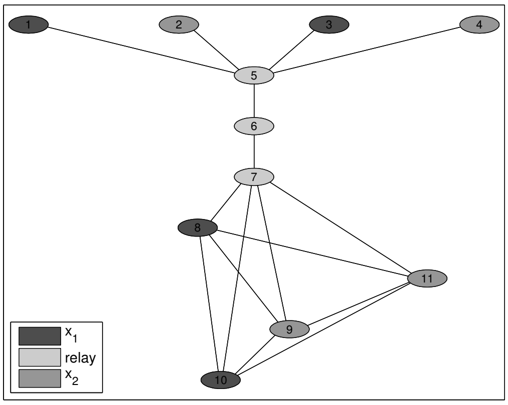

Fig. 2. Sensor network in in the second order example. Nodes in category x_1 measures the first state, relay nothing and x_2 the second state.

There are three different types of nodes, one that measures x_1, one that measures x_2 and one type that does not measure any of the states. The third category of nodes simply acts as relay stations to increase the range of the network. This setup results in the following measurement equations.

$$\begin{aligned} y_{1,3,8,10}(k) &= \begin{bmatrix} 1 & 0 \end{bmatrix} x + e_{1,3,8,10}(k) \\ y_{5,6,7}(k) &= \begin{bmatrix} 0 & 0 \end{bmatrix} x + e_{5,6,7}(k) \\ y_{2,4,9,11}(k) &= \begin{bmatrix} 0 & 1 \end{bmatrix} x + e_{2,4,9,11}(k) \end{aligned} \tag{29}$$

The process noise $v(k)$ and measurement noise $e(k)$ has covariance matrices $R_v = I_{2\times 2}$ and $R_e = I_{11\times 11}$ respectively. This implies that the measurement noise in different nodes is uncorrelated. The network topology is given by the bi-directional graph in Figure 2.

Next the weights W_{ij} and Kalman gains K_i were calculated using the algorithm described in section IV-B. The trace of the estimation error covariance P_{ii} is plotted in Figure 3. As a comparison the covariance when measurements are shared was also calculated, as described in section III. The communication constraints introduces a larger estimation error covariance as compared to the case when all nodes has access to all measurements without delay, this is apparent when these two cases are compared with the global information case in Figure 3.

From Figure 3 the conclusion can be drawn that the estimation error covariance is smaller when the graph is more connected as in the case with node 7 to 11. Nodes 1 to 4 have to communicate their estimates through node 5 which increases the variance in these nodes. Node 5 on the other hand has information from all four nodes and thus has a smaller estimation error covariance despite the fact that this node does not take any measurements itself. It is also worth noticing that the optimal solution is very close to the weighted average solution for this particular example.

Because of the size of this problem all 42, 2 × 2 weight matrices will not be explicitly given, but only a few examples. Node 1 for example has the following two

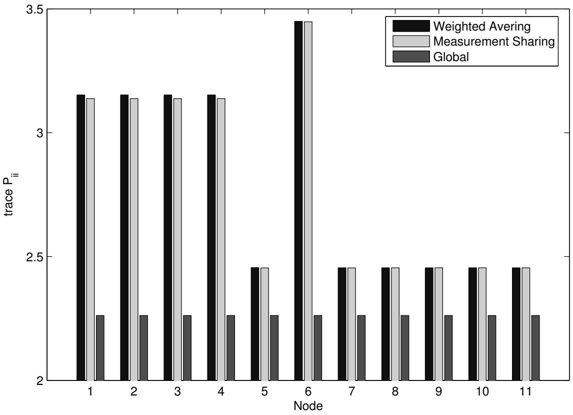

Fig. 3. Estimation error covariance in different nodes for the second order example. Global refers to a scenario where all node have instantaneous access to all measurements.

weights

$$W_{11} = \begin{bmatrix} 0.9235 & 0 \\ 0 & 0 \end{bmatrix} \qquad W_{15} = \begin{bmatrix} 0.0765 & 0 \\ 0 & 1 \end{bmatrix} \tag{30}$$

Node 1 only measures state 1 and thus all information concerning state 2 is collected from node 5, making the fourth element of W_{15} equal to 1. The estimate in node 5 contains some information about state 1 and thus the first element in W_{15} is non zero.

B. Temperature Sensing Network

Here a small temperature sensor network will be used as an example to demonstrate the algorithm. The sensor network is simulated in TrueTime which is a Matlab/Simulink-based simulator for real-time control systems. For a detailed description of TrueTime see for example [12].

The network communication was simulated using a IEEE802.11b TrueTime Wireless LAN module [13]. All nodes are transmitting with an output power of 100 mW and assumes that the network is active if the received power exceeds 1 μW. A node is allowed to retransmit lost packages 5 times before the packet is dropped. A packet is assumed to be OK if the expected raw bit error rate does not exceed 10 parts per million. The physical topology is represented by the graph in Figure 4.

Each network node is implemented in a TrueTime kernel block, which simulates a real-time kernel. The sensor task is run as a periodic task with period 0.05 s. This allows for accurate simulation of timing problems due to for example network collisions. Each node executes the following cycle each sample:

1) Analog to digital conversion of measurement.
2) Update local estimate.
3) Make estimate available for use.
4) Send local estimate to neighbors.
5) Wait for estimate from neighbors for a pre-specified time.
6) Merge estimates from neighbors.
7) Predict.

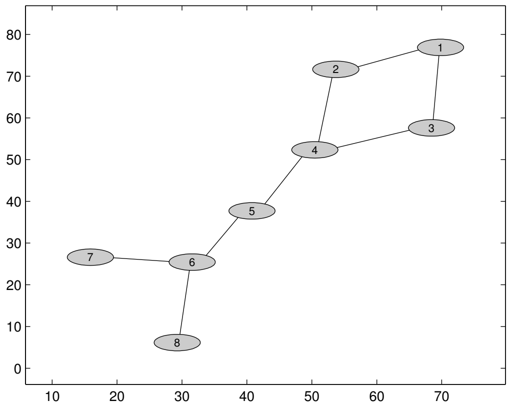

Fig. 4. Physical topology of the temperature sensing network.

If step 5 fails within the required timeout the estimate from the last successful communication is used.

As the task is only to estimate the temperature in those points in space where the sensor nodes are located a very simple temperature diffusion model is used. The emphasis here is not to model temperature diffusion, but to investigate the impact of packet loss.

The sensor network consists of 8 nodes thus the temperature is model as a 8th order discrete time linear model. Each state i represents the temperature in the proximity of node i. The state is updated according to

$$x_i(k+1) = x_i(k) + \sum_{j \in V} \frac{1}{R_{ij}} (x_j(k) - x_i(k)) \tag{31}$$

where R_{ij} is a weight depending on the distance between node i and j and V is the set of all nodes. In this example R_{ij} is the euclidean distance between node i and j. For the off-line computations of K_i and W_{ij} the noise covariance matrices where chosen as $R_w = I_{8\times 8}$ and $R_e = I_{8\times 8}$.

To investigate the degradation in estimation performance introduced by the distributed scheme, estimates from node 1,4 and 8 of state 1 was compared to the actual value of state 1. The system was simulated for 2 seconds with initial condition $x(0) = \begin{bmatrix} 40 & 20 & 20 & 20 & 20 & 20 & 20 & 20 \end{bmatrix}^T$. All nodes were initiated with a zero initial estimate.

A time plot of the three estimates together with the actual state is given in Figure 5. To be able to evaluate the impact of packet loss the number of packets received by node 2 and 3 is plotted in Figure 6. These two nodes are the only ones that experienced packet loss in this simulation. As both these nodes have 2 neighbors, the nominal number of received packets for both nodes is 2.

As expected the influence of initial conditions is greater in node 8, as all information has to travel through 4 nodes to get there. This can be observed through the slower convergence of that estimate in Figure 5.

Because there are two parallel paths from node 1 to 4, the effects of dropped packets will be most evident if both

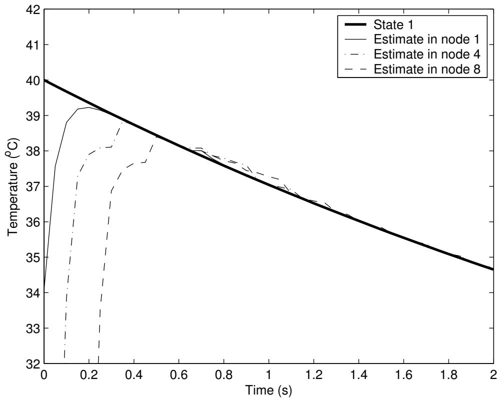

Fig. 5. Time plot of the estimates from node 1,4 and 8 of state 1 in the temperature sensor network together with the actual state.

node 2 and 3 drops packets at the same time. This happens approximately in the time intervals $[0.2\ 0.4]$ and $[0.7\ 0.9]$. As can be seen in Figure 5 the estimate in node 4 levels out in this interval, due to the fact that the old estimate from node 1 is used. This effect can also be seen in the estimate from node 8 but 0.15 s later. This corresponds to 3 sampling intervals which is the effective distance between node 4 and 8. The same effect can be seen in the time interval $[0.7\ 0.9]$.

VI. Conclusions and Future Work

In this paper an optimization based algorithm for distributed estimation was developed. The algorithm is based on standard Kalman filtering results and then extended with one step where nodes merge their estimates. The estimates are merged by a weighted average approach. The performance is then compared to a scenario where nodes share measurements. As the measurement sharing solution is computable, but practically unattractive due to a potentially high bandwidth requirement, it is possible to verify how close the weighed average solution is. For the examples presented here the maximum difference between measurement sharing and weighted averaging is less than 0.5% for all nodes.

Also the algorithm applies to a broad category of graphs, including graphs with loops. The weights are optimized off-line allowing only estimates to be communicated among the nodes. All communication is restricted to neighboring nodes, which allows the algorithm to scale.

The algorithm was applied to one second order system with 11 nodes and one temperature sensing network with 8 nodes. Simulations of the temperature sensing network made in TrueTime indicate that the algorithm is quite robust to dropped packages. The algorithm has also been tried on a larger example with 100 nodes estimating a second order system.

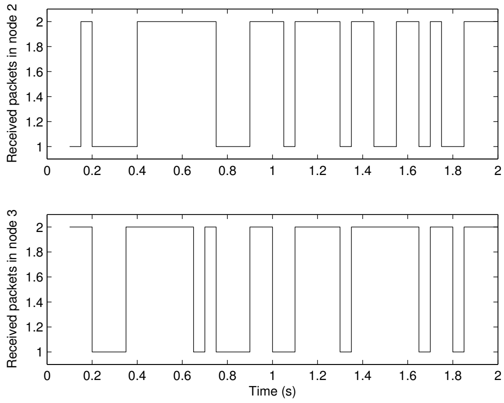

Fig. 6. Number of received packets per sample in each sample by node 2 and 3 in the temperature sensor network (The nominal number of received packages is 2).

VII. Acknowledgements

This work was partly supported by the European Union project Control and Computation, HYCON, RUNES and the Swedish Research Council.

References

[1] H. D.L. and L. J., "An introduction to multisensor data fusion," *Proceedings of the IEEE*, vol. 85, no. 1, pp. 6–23, 1997.

[2] R. Viswanathan and P. Varshney, "Distributed detection with multiple sensors part i. fundamentals," *Proceedings of the IEEE*, vol. 85, no. 1, pp. 54–63, 1997.

[3] D.-W. H.F., R. B.Y.S., and H. H., "Toward a fully decentralized architecture for multi-sensor data fusion," *Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on*, pp. 1331–1336 vol.2, 1990.

[4] S. Grime, H. F. Durrant-Whyte, and P. Ho, "Communication in decentralized data-fusion systems," in *In Proc. American Control Conference*, 1992, pp. 3299–3303.

[5] F. Bourgault and H. F. Durrant-Whyte, "Communication in general decentralized filters and the coordinated search strategy," in *The 7th International Conference on Information Fusion*, July 2004.

[6] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, no. 1, pp. 65–78, September 2004.

[7] D. P. Spanos and R. M. Murray, "Distributed sensor fusion using dynamic consensus," in *Proccedings of the 16th IFAC World Congress*, July 2005.

[8] R. Olfati-Saber, "Distributed kalman filter with embedded consensus filters," in *Proceedings of the 44th IEEE Conference on Decision and Control, and European Control Conference*, December 2005.

[9] K. Chang, T. Zhi, and R. Saha, "Performance evaluation of track fusion with information matrix filter," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 38, no. 2, pp. 455–466, 2002.

[10] S. Sun, "Multi-sensor optimal information fusion kalman filter for discrete multichannel arma signals," *Intelligent Control. 2003 IEEE International Symposium on*, pp. 377–382, 2003.

[11] Y. Bar-Shalom and L. Campo, "The effect of common process noise on two-sensor fused-track covariance." *IEEE Transactions on Aerospace and Electronic Systems*, pp. 803–805, November 1986.

[12] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K.-E. Årzén, "How does control timing affect performance?" *IEEE Control Systems Magazine*, vol. 23, no. 3, pp. 16–30, June 2003.

[13] M. Andersson, D. Henriksson, A. Cervin, and K.-E. Årzén, "Simulation of wireless networked control systems," in *Proceedings of the 44th IEEE Conference on Decision and Control and European Control Conference ECC 2005*, Seville, Spain, Dec. 2005.