Faulty node detection in a consensus network

Navid Rezazadeh

***Abstract*—This project considers the problem of designing an algorithm to diagnose a faulty node in an average consensus network. A consensus network consist of a node set and connections through which the nodes are able to communicate.The goal of a such network is for the nodes to reach the average of their initial decisions. This project deals with a situation where one of the nodes starts to behave abnormally. A framework is proposed where a node is assigned to be the network's safeguard. The safeguard node uses a Kalman filter to located the abnormal node using hypothesis testing and also to detect and compensate for the abnormal behavior.**

I. Introduction

Technological advances in ad-hoc networking and the availability of low-cost reliable computing,data storage and sensing devices have made possible scenarios where the coordination of many subsystems extends the range of human capabilities. Smart grid operations, smart transportation,smart healthcare and sensing networks for environmental monitoring and exploration in hazardous situations are just a few examples of such network operations. In these applications, the ability of a network system to, in a decentralized fashion, fuse information, compute common estimates of unknown quantities, and agree on a common view of the world is critical. These problems can be formulated as agreement problems on linear combinations of dynamically changing reference signals or local parameters. Consensus problems have attracted much attention among researchers studying distributed and decentralized automated systems. Broadly speaking, a consensus problem is one in which several spatially distributed agents or processors must reach a common output value, but without recourse to a central coordinator or global communication. This work focuses on linear consensus, i.e. consensus in which all agents must converge to a linear combination of their individual input values [1], [2]. One of the recently developed frameworks is distributed Kalman filtering. In this framework, several sensors are connected through a network in order to measure a given process an perform a Kalman filtering. In absence of a central computer the nodes are able to perform a distributed Kalman filtering to measure the states of the process. The main algorithm used in this framework is average consensus algorithm for the agents to consent of a unique estimate of the states [3]. However, these distributed networks are not robust to network faults and any malfunction in the network can lead to a corruption the final value of average consensus. The network changes can range from node removal, nodal connections to a faulty node who sends corrupted signals to other nodes in the network or a frozen node. In the cases where topology of the network is in the matter of the change, the average consensus algorithms are able to handle it. However, in situations that the nodes start to misbehave, the average consensus network are not able to adapt to this behavior and their performance degrades. This project is focused on the later, where a signal node in the network starts to misbehave by sending corrupted signals to the other node in the network. The first goal of the project is to come up with an approach that the faulty node is detected. The second goal of the project is to reveal how the faulty node has effected the the network (i.e. finding the corrupted signal sent to the network.) These goal are full filled through measuring the state of one of the nodes in the network. The justification of this assumptions come from the fact that in real world application it may not be possible to measure states of all nodes directly and only some nodes that are accessible.

The authors are with the Department of Mechanical and Aerospace Engineering, University of California Irvine, Irvine, CA 92697, {nrezazad}@uci.edu.

II. Preliminaries

Graph theory: in the following, we review some basic concepts from algebraic graph theory following [4]. An un-directed graph is a triplet $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$, where $\mathcal{V} = \{1, \dots, N\}$ is the *node set*, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the *edge set* and $\mathbf{A} = [\mathsf{a}_{ij}] \in \mathbb{R}^{N \times N}$ is a *adjacency* matrix with the property that $\mathsf{a}_{ij} = 1$ if $(i, j) \in \mathcal{E}$ and $\mathsf{a}_{ij} = 0$, otherwise. In an un-directed graph if $\mathsf{a}_{ij} = \mathsf{a}_{ji}$ for all $i, j \in \mathcal{V}$. An edge from i to j, denoted by (i, j), means that agent j can send information to agent i and vice versa. For an edge $(i, j) \in \mathcal{E}$, i is called an *neighbor* of j. We denote the set of the neighbors of an agent $i \in \mathcal{V}$ by $\mathcal{N}^i$. A *path* is a sequence of nodes connected by edges. A digraph is called *strongly connected* if for every pair of vertices there is a directed path connecting them. We refer to a strongly connected and undirected graph as a *connected graph*. The *degree* of a node i, are respectively, $\mathrm{d}^i = \sum_{j=1}^{N} \mathsf{a}_{ji}$. The *Laplacian* matrix is $\mathbf{L} = [\ell_{ij}]$ is $\mathbf{L} = \mathbf{I} - (\mathbf{D} - \mathbf{A})\delta$, where $\mathbf{D} = \mathrm{Diag}(\mathrm{d}^1, \cdots, \mathrm{d}^N) \in \mathbb{R}^{N \times N}$ and δ is constant chosen is was way that Laplacian $\mathbf{L}$ has one zero eigenvalue $\lambda_1 = 1$ and the rest of its are $0 \leq \lambda_i < 1$. Also we define the $\mathbf{L}'$ to the $\mathbf{L}$ with the diagonal entries set to 0. Such definition of $\mathbf{L}$ results in a doubly stochastic matrix where $\mathbf{1}^\top \mathbf{L} = \mathbf{1}^\top$ and $\mathbf{L}\mathbf{1} = \mathbf{1}$

III. Problem formulation

Consider the static average consensus algorithm

$$x^i(k+1) = -\sum_{j=1}^{N} (\ell_{ii} x^i(k) + \ell_{ij} x^j(k)), \quad x^i(0) = \mathsf{r}^i, \quad i \neq j \tag{1}$$

over a strongly connected and weight-balanced digraph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathbf{A})$. For such an interaction typology, x^i of each agent $i \in \mathcal{V}$ converges to $\frac{1}{N} \sum_{j=1}^{N} \mathsf{r}^j$ as $t \to \infty$ [5]. The aggregated of the equation above is given as

$$\mathbf{x}(k+1) = \mathbf{L}\mathbf{x}(k)$$

Now we assume that a node in the system $p \in \mathcal{V}$ starts to malfunction such that the signals transmitted from it are a result of an additive Gaussian zero mean noise with covariance matrix given by $\mathbf{Q}$ at time step $k_1 < k < k_2$. Therefor, Assuming that the transmitted signal from a node at time $k_1 < k < k_2$ is given by $y^i(k)$ then we have

$$\begin{aligned} y^i(k) &= x^i(k), \quad i \neq p \\ y^p(k) &= x^p(k) + w(k), \quad w(k) \sim \mathcal{N}(0, \mathbf{I}). \end{aligned}$$

Hence the neighboring nodes of p given as $\mathcal{N}^p$ update their state based on noise measurement. Therefor the update equation at time step $k_1 < k < k_2$ becomes

$$x^i(k+1) = -\sum_{j=1}^{N}(\ell_{ii}x^i(k) + \ell_{ij}y^j(k)), \quad x^i(0) = \mathsf{r}^i, \quad i \neq j \tag{2}$$

The collective form of the equation above for time $k_1 < k < k_2$ can be written as

$$\mathbf{x}(k+1) = \mathbf{L}\mathbf{x}(k) + \mathbf{L}'\mathbf{C}^p w(k) \tag{3}$$

where $\mathbf{C}$ is a column vector with all entries of zero except the p^{th} equal to 1. the term $\mathbf{L}'\mathbf{C}^p$ sifts out the p^{th} column of $\mathbf{L}'$ This means that all the nodes who the immediate neighbors of node p are effected by the noise transmitted by the node. To see how this noise will effect the consensus value of the algorithm, one can right-multiply equation (3) with a transpose of vector of ones $\mathbf{1}^\top$. Since by definition we have $\mathbf{1}^\top\mathbf{L} = \mathbf{1}^\top$ therefor

$$\sum_{j=1}^{N} x^i(k+1) = \sum_{j=1}^{N} x^i(k) + \mathbf{1}^\top\mathbf{L}'\mathbf{C}^p w(k).$$

It means that at each step $\mathbf{1}^\top\mathbf{L}'\mathbf{C}^p w(k)$ is added to $\sum_{j=1}^{N} x^i(k)$ resulting in

$$\sum_{j=1}^{N} x^i(k+1) = \sum_{j=1}^{N} r^i(0) + \mathbf{1}^\top\mathbf{L}'\mathbf{C}^p \sum_{l=k_1}^{k_2} w^i(l).$$

for $k > k_2$. Since we assume that after time k_2 no noise is added to the network then therefore we can conclude that

$$x^i(k) = \frac{1}{N}\sum_{j=1}^{N} \mathsf{r}^j + \frac{1}{N}\mathbf{1}^\top\mathbf{L}'\mathbf{C}^p \sum_{l=k_1}^{k_2} w^i(l)$$

as $k \to \infty$.

To visualized the negative effect of additive noise, an example of a network with 7 nodes is considered 1. In this network each agents start with initial condition given by $\mathbf{x}(0) = [-3, -2, -1, 0, 1, 2, 3]^\top$. at time step 21 node 7 starts to malfunction and inject zero mean noise with variance of 1 to the network for 100 time steps.

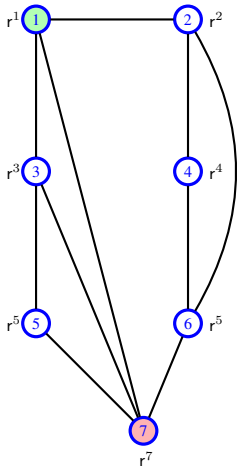

Fig. 1: A network of the nodes where the node 7 injects noise with $\mathcal{Q} = 1$ injects noise to the network through transmission.

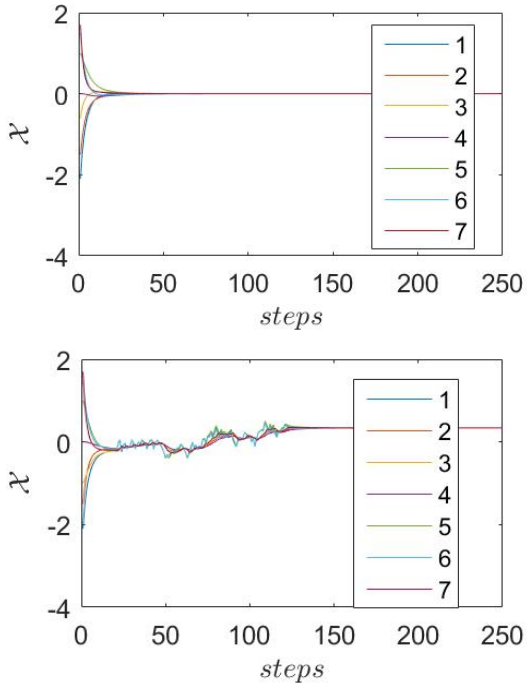

Fig. 2: each agents start with initial condition given by $\mathbf{x}(0) = [-3, -2, -1, 0, 1, 2, 3]^\top$. at time step 21 node 7 starts to malfunction and inject zero mean noise with variance of 1 to the network for 100 time steps.

Figure 2 shows misbehavior of node 7 results in a deviation of the consensus value from the designated value of 0.

Problem Statement: In the presence of adverse effect of additive noise to system on the value of the average consensus, and being able to measure the state of a safe guard node with Gaussian zero mean measurement noise (without loss of

generality node 1),

$$\mathbf{x}(k+1) = \mathbf{L}\mathbf{x}(k) + \mathbf{L}'\mathbf{C}^p w(k) \quad (4)$$
$$z(k+1) = \mathbf{H}\mathbf{x}(k+1) + v(k+1), \quad v \sim \mathcal{N}(0, \mathbf{R}), \quad (5)$$

with $\mathbf{H} = [1, 0, ..., 0]^\top$ and fairly small R (This assumption is justify able since in reality the measurement is to read a nodes state from a computer whose accuracy comes to the number of significant digits that the computer can hold).we want to design an algorithm where it is able to (listed as the matter of importance)

- find the faulty node p.
- find the the time step that the faulty node injects noise into the network, i.e. k_1 and k_2
- find the sequence of added noise $w(k)$, $k_1 < k < k_2$
- find the variance value $\mathbf{Q}$ with knowing that $\mathbf{Q} \leq \tilde{\mathbf{Q}}$

IV. Kalman Filtering Scheme

To solve the proposed problem, we make use of a Kalman filtering scheme given as

Time Update:

$$\bar{\mathbf{x}}(k+1) = \mathbf{L}\hat{\bar{x}}(k)$$
$$\bar{\mathbf{P}}(k+1) = \mathbf{L}\hat{\mathbf{P}}(k)\mathbf{L} + \mathbf{Q}$$

Measurement Update:

$$\boldsymbol{\nu}(k+1) = z(k+1) - \mathbf{H}\bar{\mathbf{x}}(k+1)$$
$$S(k+1) = \mathbf{H}(k+1)\bar{\mathbf{P}}(k+1)\mathbf{H}^\top(k+1) + R$$
$$\mathbf{W}(k+1) = \bar{\mathbf{P}}(k+1)\mathbf{H}^\top S^{-1}(k+1)$$
$$\hat{\mathbf{x}}(k+1) = \bar{\mathbf{x}}(k+1) + \mathbf{W}(k+1)\nu(k+1)$$
$$\hat{\mathbf{P}}(k+1) = \bar{\mathbf{P}}(k+1) - \mathbf{W}(k+1)S(k+1)\mathbf{W}^\top(k+1)$$

with assumption that safeguard who measures the state of node 1 does not have any specific knowledge about the initial conditions of the nodes r^i of the network which leads to initializing the kalman filter with $\hat{\mathbf{P}}(0) = 100\mathbf{I}$ and $\hat{\mathbf{x}}(0) = \mathbf{0}$.

To answer the first question about finding the faulty node p, the safeguard can run the Kalman filter several times for different possibilities of $\mathbf{C}^p$ (e.g. $\mathbf{C}^{p=1} = [1, 0, ..., 0]^\top$, $\mathbf{C}^{p=2} = [0, 1, ..., 0]$) and choose $\mathbf{C}^p$ such that likelihood of the filter is maximized or equivalently choose the model which minimizes

$$\sum_{k=1}^{K} \nu(k)S^{-1}(k)\nu(k)$$

Since the measurement variance R is assumed to be fairly small, then the safeguard can set $\mathbf{Q} = \tilde{\mathbf{Q}}$ for all time steps. The Kalman filter will pick up measurements and build its results based on it naturally since the measurements are assumed to be fairly accurate. Building on the introduced example on figure 2, running 7 Kalman filters with the described setup, give the value of 1.79×10^3, 220, 1.79^4, 4.06×10^3, 3.87×10^3 and 102 where clearly we can conclude that node 7 is the culprit of noise injection.

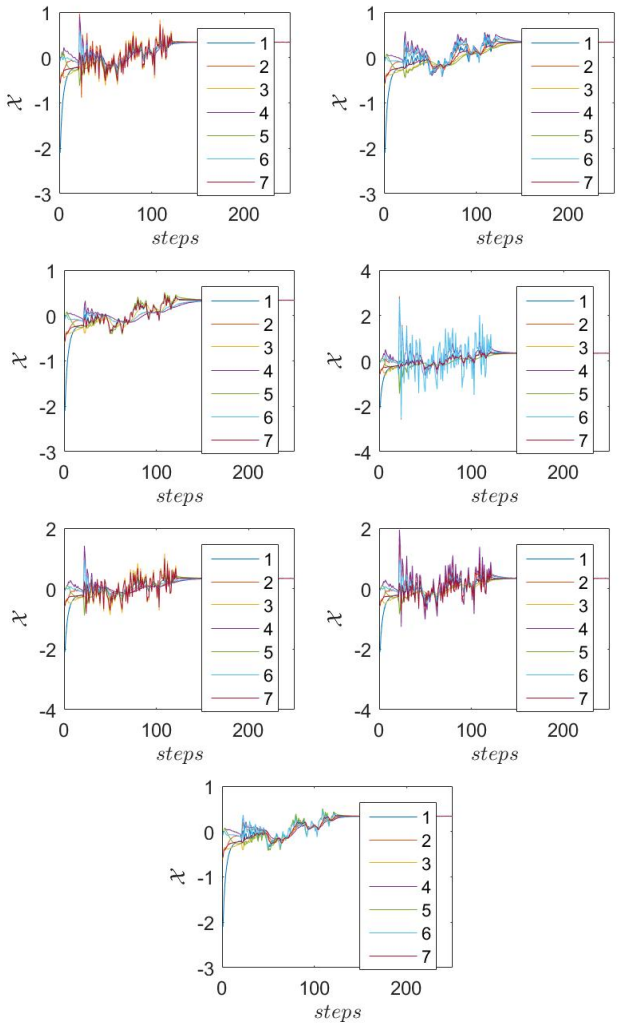

Fig. 3: running 7 Kalman filters with the described setup, give the value of 1.79×10^3, 220, 1.79^4, 4.06×10^3, 3.87×10^3 and 102 where clearly we can conclude that node 7 is the culprit of noise injection.

Figure 3 shows the estimated estate trajectories for different hypothesises on the node injecting the noise to its communicated signal.

To find the time k_1 and k_2 where the noise was injected by the culprit node and also the sequence value of the noise, we can use equation (4) and conclude that

$$\hat{\mathbf{x}}(k+1) - \bar{\mathbf{x}}(k+1) = \mathbf{L}'\mathbf{C}^p w(k)$$

Since $\mathbf{L}'\mathbf{C}^p$ is a tall matrix (more rows than column) the least square estimate of $w(k)$ becomes

$$\hat{w}(k) = (\mathbf{C}^{p\top}\mathbf{L}'^\top)^\top[\mathbf{C}^{p\top}\mathbf{L}'^\top\mathbf{L}'\mathbf{C}^p]^{-1}(\hat{\mathbf{x}}(k+1) - \bar{\mathbf{x}}(k+1))$$

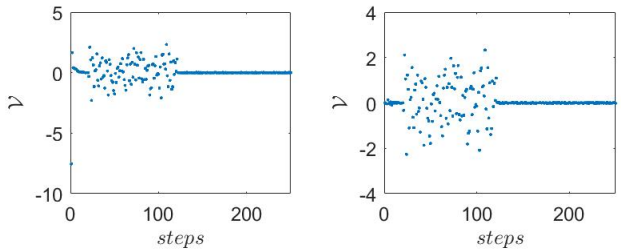

Fig. 4: The estimated noise $w(k)$ assuming the hypothesis of node 7 adding noise to the system. The right figure is the estimated noise $w(k)$ using the zero initial condition, however the left figure is shows the noise $w(k)$ estimated using initial condition set to $\mathbf{x}^\star(0)$

As an example we can perform the noise sequence estimation on the numerical example using the validated hypothesis.

Figure 4 shows the estimated noise sequence. It can be noted that the start time and stop time of noise injection in clear from the graph. However the starting time of simulation, has some noise components to it, which is from the fact that the starting point $\hat{\mathbf{x}}(k)$ is set to be zero and it takes time for the filter to converge to the real trajectory of $\mathbf{x}(k)$. To overcome this issue we can perform a smoothing backward as

$$\begin{aligned}\mathbf{x}^\star(k) =&\hat{\mathbf{x}}(k) + \mathbf{P}(k)\mathbf{L}^\top\bar{\mathbf{P}}(k+1)^{-1}(\mathbf{x}^\star(k+1) - \bar{\mathbf{x}}(k+1))\\ \mathbf{P}^\star(k) =&\mathbf{P}(k) - \mathbf{P}(k)\mathbf{L}^\top\bar{\mathbf{P}}(k+1)^{-1}[\bar{\mathbf{P}}(k+1)\\ &- \mathbf{P}^\star(k+1)]\bar{\mathbf{P}}(k+1)^{-1}\mathbf{L}\mathbf{P}(k)\end{aligned}$$

and use $\mathbf{x}^\star(0)$ as the initial condition of Kalman Filtering. Figure 4 shows the estimated noise $w(k)$ using this setup and as it is noted the wrongly estimated noise at early steps is removed.

The next step is to estimate the variance of the process noise $\mathbf{Q}$. Since the value of the process noise is estimated and there is a clear edge to start and stop point an empirical estimate of variance is given as

$$\hat{\mathbf{Q}} = \frac{1}{k_2 - k_1}\sum\nolimits_{j=k_1}^{k_2}(\bar{w}(j))^2$$

For the given numerical example this value is given as 1.05 which is a rational estimate of the real value which is 1.

Remark IV.1. *It is possible to consider $\mathbf{C}^p$ as an unknown state of the dynamics* (4) *augment the dynamics as*

$$\begin{aligned}\mathbf{x}_a(k+1) = \begin{bmatrix}\mathbf{x}(k+1)\\ \mathbf{C}^p(k+1)\end{bmatrix} = \begin{bmatrix}\mathbf{L} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{x}(k)\\ \mathbf{C}^p(k)\end{bmatrix} + \begin{bmatrix}\mathbf{L}'\mathbf{C}^p(k)\\ \mathbf{0}\end{bmatrix}\\ z_a(k+1) = \mathbf{H}_a\mathbf{x}_a(k+1) + v(k+1)\end{aligned}$$

which is a nonlinear equation in terms of the states. Lionization of the dynamics yields

$$\mathbf{L}_a = \begin{bmatrix}\mathbf{L} & \mathbf{L}'\\ \mathbf{0} & \mathbf{I}\end{bmatrix}[$$

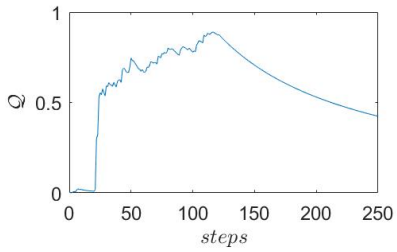

Fig. 5: The averaged empirically estimated $Q(k)$.

and it can be shown that the pair $(\mathbf{L}_a, \mathbf{H}_a)$ is not observable, therefor it is not possible to follow this approach to estimate $\mathbf{C}^p(k)$.

Adaptive Estimation of variance : Using the measurement it is possible to calculate the estimated variance $\mathbf{Q}$ given the innovation terms at each time step as

$$\hat{\mathbf{Q}}_k = \mathrm{E}[\hat{w}_{k-1}\hat{w}_{k-1}^\top]$$

given that $\hat{w}(k) = (\mathbf{C}^{p\top}\mathbf{L}'^\top)^\top[\mathbf{C}^{p\top}\mathbf{L}'^\top\mathbf{L}'\mathbf{C}^p]^{-1}(\hat{\mathbf{x}}(k+1) - \bar{\mathbf{x}}(k+1))$ and the fact that $\hat{\mathbf{x}}(k+1) = \bar{\mathbf{x}}(k+1) + \mathbf{W}(k+1)\nu(k+1)$, then we get

$$\begin{aligned}\hat{\mathbf{Q}}_k = [(\mathbf{C}^{p\top}\mathbf{L}'^\top)^\top[\mathbf{C}^{p\top}\mathbf{L}'^\top\mathbf{L}'\mathbf{C}^p]^{-1}\mathbf{W}(k)]\\ S(k)\\ [(\mathbf{C}^{p\top}\mathbf{L}'^\top)^\top[\mathbf{C}^{p\top}\mathbf{L}'^\top\mathbf{L}'\mathbf{C}^p]^{-1}\mathbf{W}(k)]^\top\end{aligned}$$

Since the estimation is at each time step we can perform an averaging to smooth the estimated variance and insert it in the filter at each time step.

$$\bar{\mathbf{Q}}_k = 0.9 * \bar{\mathbf{Q}}_{k-1} + 0.1 * \hat{\mathbf{Q}}_k$$

Remark IV.2. *Instead of using the calculated $\hat{\mathbf{Q}}_k$ using the innovation term, it is possible to calculate the emprical variance and smooth it.*

$$\hat{\mathbf{Q}}_k = \frac{1}{k}\sum\nolimits_{j=i}^{k}(\bar{w}(j))^2$$

REFERENCES

[1] D. P. Spanos, R. Olfati-Saber, and R. M. Murray, "Dynamic consensus on mobile networks," in *IFAC world congress*, pp. 1–6, Citeseer, 2005.

[2] S. S. Kia, B. Van Scoy, J. Cortes, R. A. Freeman, K. M. Lynch, and S. Martinez, "Tutorial on dynamic average consensus: the problem, its applications, and the algorithms," *arXiv preprint arXiv:1803.04628*, 2018.

[3] R. Olfati-Saber, "Distributed kalman filtering for sensor networks," in *2007 46th IEEE Conference on Decision and Control*, pp. 5492–5498, IEEE, 2007.

[4] F. Bullo, J. Cortés, and S. Martínez, *Distributed Control of Robotic Networks*. Applied Mathematics Series, 2009.

[5] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," vol. 95, no. 1, pp. 215–233, 2007.