Agents Misbehaving in a Network: A Vice or a Virtue?

Iman Shames, André M. H. Teixeira, Henrik Sandberg, Karl H. Johansson,
Royal Institute of Technology

Abstract

Misbehaviors among the agents in a network might be intentional or unintentional, they might cause a system-wide failure or they might improve the performance or even enable us to achieve an objective. In this article we consider examples of these possible scenarios. First, we argue the necessity of monitoring the agents in a network to detect if they are misbehaving or not and outline a distributed method in which each agent monitors its neighbors for any sign of misbehavior. Later, we focus on solving the problem of distributed leader selection via forcing the agents to temporarily misbehave, and introduce an algorithm that enables the agents in a network to select their leader without any interference from the outside of the network.

Recent years have witnessed a more pronounced presence of robotic networks in our day-to-day life. Naturally, as the domain of activity of these systems gets closer to our daily life one cannot ignore the possibility of a scenario similar to the one described by Isaac Asimov in his masterpiece, "I, Robot," and stop wondering what might happen if these metallic servants stop following the rules they are supposed to follow and express new behaviors, that possibly can be labelled as misbehaviors?[1]

The answer to this question is not straightforward. Consider the scenario depicted in Fig. 1 where the robot R_1, as a member of a group of networked robots, has the goal to carry the cargo from point A to point B such that the time spent is minimized. This is done via running a simple graph search algorithm on the environment map uploaded to the memory of R_1. The path for all the robots is depicted in Fig. 1a. However, due to different factors, such as minimum turning radius, narrowness of the passage, congestion, etc., this path does not correspond to an optimum path when the overall time used by each of the robots is taken into account. In this scenario, after a time R_1 modifies its path as depicted in Fig. 1b which does not clearly correspond to a minimum time path and is against the original programming of R_1. This misbehavior results in an improved performance, for example because R_1 does not contribute to the congestion and hence it helps both itself and the other robots to achieve their tasks in a more timely manner. Now, consider the other case depicted in Fig. 2. In this scenario a group of robots are supposed to move in unison with equal velocities from the left side of the environment to the right side while forming a desired shape. Figure 2a depicts the case where they achieve this objective and agree on a common velocity to achieve the task. However, Fig. 2b demonstrates another situation where one of the robots does not follow the rule to reach the agreement on the velocity and consequently inhibits the group from maintaining its desired shape and as a result fails to achieve its objective.

As it can be seen from the examples presented above, while an agent misbehaving in a network does not necessarily result in catastrophe, one cannot discard the importance of the detection of the situations where an agent is misbehaving, that is, not following the original programming and the rules that were defined for it. Moreover, putting aside the fact that a misbehavior may result in mayhem or just improve the efficiency of the system, we advocate that in particular scenarios certain objectives can be achieved if agents do misbehave.

Specifically, in this article we first argue how it is possible to design a distributed mechanism for the agents to monitor and be monitored by their neighbors for possible misbehaviors to prevent undesirable outcomes. These are the scenarios where a misbehavior is potentially a vice. Later, we look at a scenario where certain misbehaviors are in fact encouraged among the agents to be able to achieve a distributed task. The virtuous nature of such misbehaviors will be more apparent later.

Agents and Networks Model

Throughout this article, instead of machines or robots we use the more generic term *agent*. An agent can be a mobile robot moving around in an environment or any other piece of electric machinery, e.g., a power generator in a power grid. We assume that we deal with N agents that are labeled from 1 to N. Moreover, we assume each of the agents is associated with a set of states. These states can be the position of the agent in the case of it being a mobile robot, or its phase, in the case where it is a generator. Let $x_i \in \mathbb{R}^n$ denote the state of agent i. We further assume that the interaction among the agents can be modelled via a network, and this network in turn can be modelled with a graph $\mathcal{G}$ with vertex set $\mathcal{V} = \{1, \ldots, N\}$ and the edge set ε where the edge $\{i, j\}$ is in E if agents i and j

[1] *In this article we use the word "misbehavior" in an objective sense. Any behaviour exhibited by a machine that is not taken into account at the time of programming, regardless of its favourable or unfavourable outcome, is considered to be a misbehavior.*

 0890-8044/12/$25.00

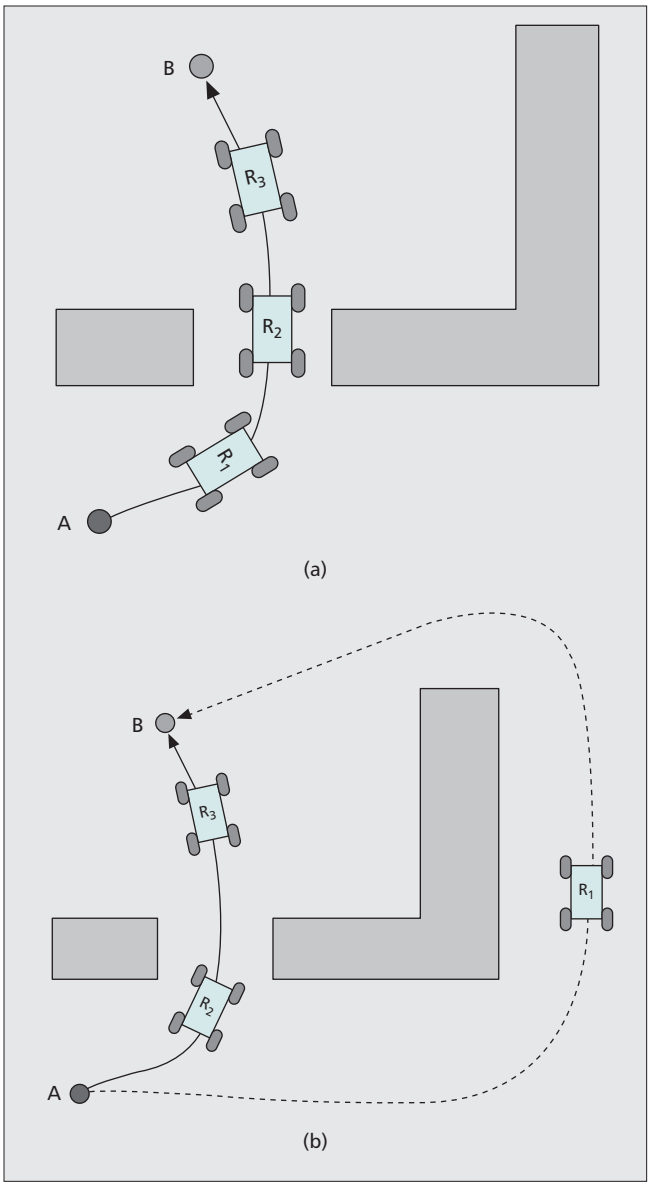

Figure 1. *a) All the agents behave according to their predefined programming, following a minimum distance yet congested path; b) one of the agents takes a longer trajectory that in the end improves the performance of the overall system by reducing congestion, thus decreasing the total time required for the task.*

directly interact. We call each agent j that interacts directly with agent i one of its neighbors, and we denote the set of all of the neighbors of i by N_i. More precisely, we assume that the state of i evolves under the following equation

$$\dot{\mathbf{x}}_i(t) = \mathbf{R}_i(\mathbf{x}_i(t), \mathbf{x}_{i1}(t), \ldots, \mathbf{x}_{i|N_i|}(t)) = \mathbf{R}_i(\mathbf{y}_i(t)) \quad (1)$$

where $i_j \in N_i$, and $\mathbf{y}_i(t)$ is all the measurements available to an agent i from itself and its neighbors. As a result the whole network can be described by

$$\dot{\mathbf{x}}(t) = \mathbf{R}(\mathbf{x}(t))$$

where $\mathbf{x}(t) = [x_1^T(t), \ldots, \mathbf{x}_N^T(t)]^T$ and $\mathbf{R}(\cdot) = [\mathbf{R}_1^T(\cdot),\ldots,\mathbf{R}_N^T(\cdot)]^T$.

Detecting Misbehaving Agents

Several technical applications such as transport systems and industrial processes have strict safety and reliability requirements. Modes of operation not satisfying these requirements may cause great damage to both the process and its surroundings. Monitoring the operating conditions through fault detection and isolation (FDI) is of utmost importance and there exists a vast collection of methods to perform failure diagnosis. Model-based fault diagnosis is a class of methods that is of particular interest for dynamical systems. One such method, observer-based FDI, is the focus of this section.

Most of the available literature on model-based FDI focus on centralized systems where the FDI scheme has access to all the available measurements and the objective is to detect and isolate faults occurring in any part of the system [1]. Some recent work has been done on the design of distributed FDI scheme. In [2], a bank of decentralized observers is built where each observer contains the model of the entire system and receives both measurements from the local subsystem and information transmitted from other observers. A similar approach is taken in [3] where the observers communicate with each other, but they only possess models of their respective local subsystems. A mixing procedure is used to reconstruct the state of the overall system from the local estimates. Recently distributed FDI schemes for a network of linear time invariant interconnected systems were considered [4, 5].

Power networks can be viewed as large-scale and spatially distributed systems. Being a critical infrastructure, they possess strict safety and reliability constraints. Monitoring the state of the system is essential to guarantee safety and currently this is typically done in a centralized control center through a single state estimator. Due to the low sampling frequency of the sensors in these system a steady state approach is taken. These methods only allow for an over-determined system to ensure reliability. Furthermore, faults, traditionally, are handled mainly by central bad data detection schemes, see [7], and until recently the transient states of the system has not been taken into account for detecting anomalies. In recent years, measurement units with higher sampling rate have been developed, opening the way to dynamic state estimators and observer-based fault detection schemes taking in account the dynamics of the system. Some centralized FDI schemes have been proposed in the recent literature, see [7]. However, to the best of knowledge of the authors, other than the result in [5], no other distributed method has been proposed to carry out FDI in power networks, despite the inherent decentralized nature of such networks.

Control and coordination of formations of autonomous robots is another application area that requires distributed fault detection and isolation methods. The decentralized nature of these systems have rendered them highly favorable to accomplish some tasks in an efficient manner. One of the most studied tasks is to make all the nodes to reach a common point, so called, *consensus*. The consensus variable can be the physical positions of the nodes, their headings, or the relative clock biases between each pair of nodes. The interaction law that ensures the achievement of consensus is often called a consensus protocol or a consensus algorithm. Most of the distributed coordination control laws proposed in the literature, such as the consensus protocols, are not robust, in the sense that a single node failing to update its values according to the law leads to that none of the nodes are reaching the consensus in the original sense.

It is imperative to either have algorithms that are robust to the effects of possible malfunctions in the system or have monitoring schemes to observe the behavior of each node in the system and detect if any of them are not following the protocol. Since these systems are highly decentralized, the monitoring should take place in a distributed way. This section focuses on distributed FDI for a network of interconnect-

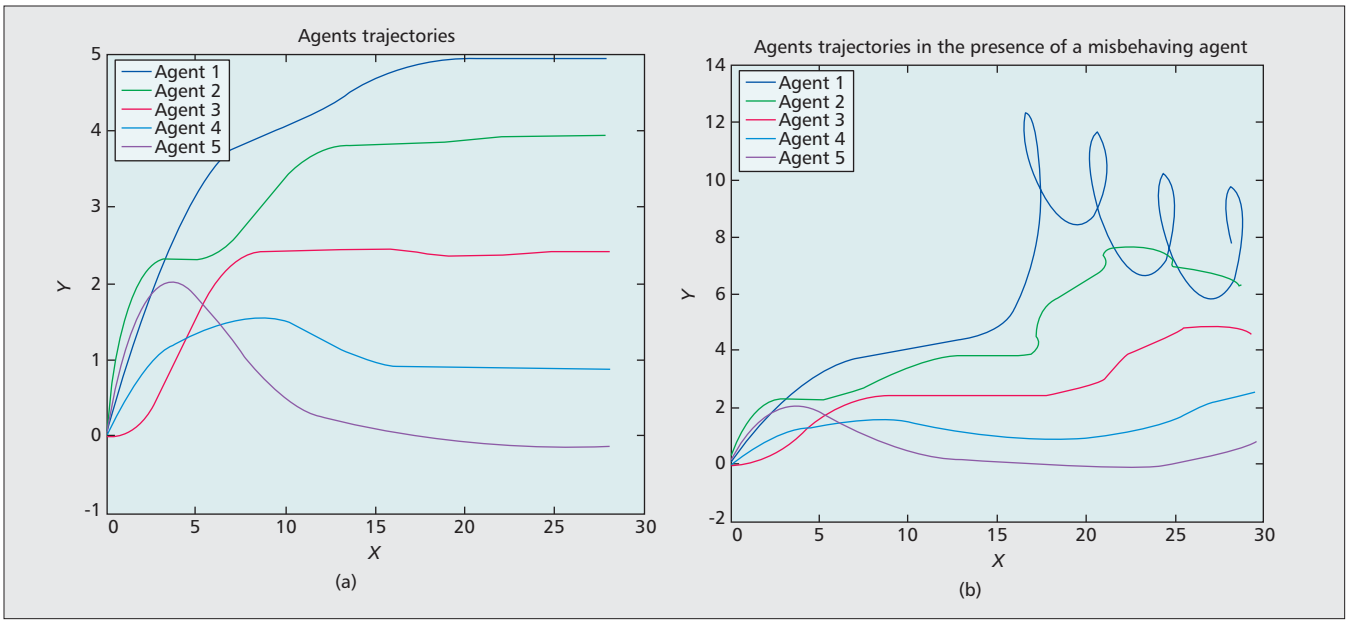

Figure 2. *a) Agents trajectories while moving in a formation; b) agents trajectories while moving in a formation in the presence of a misbehaving agent.*

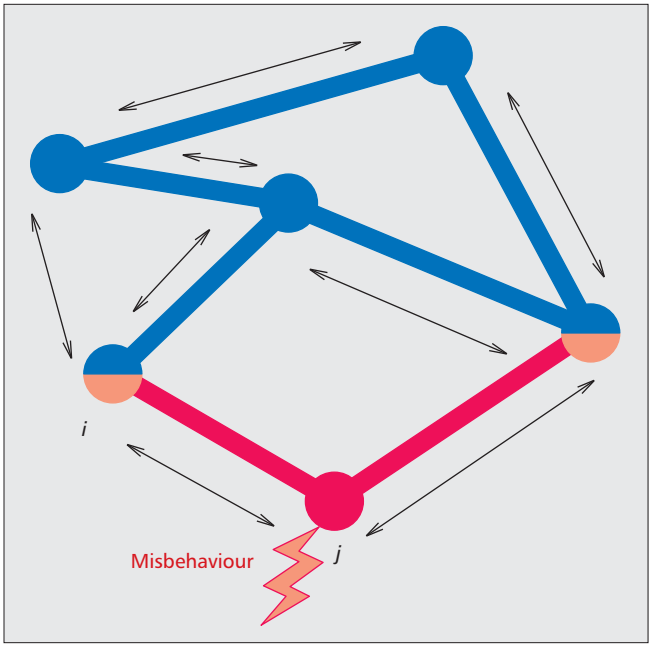

Figure 3. *A network with a misbehaving agent.*

ed dynamical systems running a distributed control law that can be seen as a general form of an inter-agent interaction law. Related to our contribution, in [8] the notion of robustness in the systems of interconnected nodes in the presence of faulty nodes in the network is introduced and formalized.

We start this section by an example. Consider the network depicted in Fig. 3 where each agent i is governed by an equation similar to the one given in Eq. 1. At a time t_0 a fault occurs at j that is one of the neighbors of i. For the purposes of this article we consider an agent j to be faulty if its states are not governed by

$$\dot{\mathbf{x}}_j(t) = \mathbf{R}_j(\mathbf{y}_j(t)) + f_j(t)$$

where $f_j(t)$ is a fault signal. We are interested in addressing the following problem.

Problem 1 (Distributed Fault Detection and Isolation): How can each agent i in the network detect a fault in the network and isolate the faulty agent j, if j is a neighbor of i?

To achieve the fault detection and isolation task, first, we present the following definition.

Definition 1: A residual $r_i^k(t)$ is a fault indicator function that is calculated at node i and satisfies

$$||r_i^k(t)|| = 0 \Leftrightarrow ||f_j(t)|| = 0 \quad \forall j \neq k \in \mathcal{N}_i.$$

Now, if agent i can calculate residuals that satisfy the condition of Definition 1 using only local measurements $\mathbf{y}_i$ and $\mathbf{R}(\cdot)$, then by applying Algorithm 1 a faulty node j can be detected by all the nodes in $\mathcal{N}_j$. However, all the other nodes in the network only detect the existence of a fault in the network and the exact identity of the faulty node is unknown to them. For a certain class of networked systems with a linear $\mathbf{R}(\cdot)$ it is shown that such residuals can be generated using unknown input observers. For more details see [5].

Remark 1: If a node j is faulty, all the nodes in the network detect that there exists a faulty node in the network unless the fault signal happens to coincide with a system zero. However, only the nodes in the one-hope neighborhood of j can isolate node j as the faulty node in the network.

We conclude this section with an example. Consider the power network depicted in Fig. 4 as a network of interconnected machines. The power network is evolving towards the steady-state when, at time instant $t = 6s$, a fault occurs at agent (node) 5. By implementing a bank of observers at agent 7, the fault is successfully detected and isolated, the residuals calculated at agent 7 are presented in Fig. 5.

Employing Misbehaving Agents to Select a Leader in A Network

To accomplish some of the tasks in networked systems it is required that one agent or a subset of them act as leaders of the group. For example, an agent should act as reference clock, in clock synchronization (This leader agent "knows" the global time of the network.), or an agent's heading should be considered as the reference heading when the agents are

for all the agents in the neighborhood of *i* **do**
  Generate a residual.
end for
if For one of the neighbors, *j*, the magnitude of its corresponding residual is less than a threshold and for all the rest of the neighbors the corresponding residuals magnitudes are more than the threshold. **then**
  Node *j* is faulty.
else if All the residuals associated with the neighbors are more than a predefined threshold. **then**
  There exists a faulty node in the network that is not a neighbor of i.
else if All the residuals associated with the neighbors are less than a predefined threshold. **then**
  There is no faulty node in the network.
end if

Algorithm 1. *D-FDI of faulty nodes at node* i.

accomplishing flocking (This agent knows what the desired direction of motion for the network is.). Such behaviors are observable in the animal kingdom as well [9], e.g. the alpha male in a pack of wolves determines where the pack should head for accomplishing foraging activities, see [10]. The role and importance of leaders in the multi-agent systems are well-known as well [11], and in many scenarios the presence of a leader seems to be crucial for achieving the desired objective, [12].

Since success in accomplishing tasks depends on choosing a leader, one may ask how this leader is being selected in multi-agent systems. There are two different ways readily available in the literature to answer this question. The first one is that the roles of the agents in the system and their relation to the other agents can be explicitly assigned to each of them. The second way is via a distributed algorithm known as `FLOODMAX`, [12]. While this second way is distributed, it assumes that the agents can easily communicate with each other and pass messages to other agents in the network. However, if one revert to animal kingdom to seek an answer to this question, she may find that the roles in groups of animals are neither assigned explicitly nor any voting-like mechanism is involved to choose a leader. The way that animals accomplish leader selection is via observing each others' behaviors, and usually the strongest one is acknowledged as the dominant member of the group (leader), [13]. This section is an endeavour to answer the aforementioned problem in the context of multi-agent network in a way similar to that of animals. The results of this article are based on [14] where the idea of selecting a leader in a network of dynamical systems was proposed for the first time.

Enabling a formation of autonomous agents to choose a leader amongst themselves is very important when one is interested in designing robust networked systems. Consider the case where only a subset of robots in a robotic network have the computational and sensory resources to calculate a collision free path for the formation of the robots in an environment. Initially, one of these robots is designated as the leader and drives the formation along a desired path. However, halfway, due to a fault or an attack it becomes incapable of controlling the trajectory of the formation. To complete the task a leader responsible for trajectory planning and control should be selected from the remaining "capable" robots. This can be achieved via the implementation of a distributed leader selection algorithm akin to that proposed in this article.

Now we are ready to pose the problem that we consider in this section.

Problem 2 (Distributed Leader Selection): Consider a network of N agents under the interconnection law in Eq. 1. Moreover, assume that the label of each agent is not known to the others. How can one and only one leader be selected without having neither explicitly assignment of the roles to the agents nor direct communication between the agents?

The basic idea in addressing the aforementioned problem is that each agent i either bids for leadership or makes a decision at the beginning of certain time intervals, or "time slots". By bidding for leadership or equivalently, taking a bidding action, we mean that it modifies its local interaction law to be

$$\dot{\mathbf{x}}_i(t) = \mathbf{R}_i(\mathbf{y}_i(t)) + \ell_i(t). \tag{4}$$

where $\ell_i(t)$ is different from $\ell_j(t)$ for all $j \neq i$, and is called agent i leadership action. The lengths of these time slots are chosen in a way to allow for the agents to observe the states of the system after a transient time. We consider the time slots to be of length $2T$. During each of the time slots each agent i generates a residual, $r_i(t)$ considering that the only agent, if any, bidding for leadership is i itself. If at time t_0, that is end of a time slot, $r_i(t_0) = 0$, then the agent i infers that no other agent has bid for leadership for the time $t \in [t_0 - 2T, t_0]$.

At time $t = 0$, each potential leader agent i chooses a random uniformly distributed integer number between 1 and –n $\in \mathbb{Z}_{\geq 0}$, $\tilde{t}_i$, corresponding to the number of the time slot that it waits until, at the beginning of which the agent starts bidding for leadership, i.e. $\bar{\tau}_i = 2\tilde{t}_i T$, in case it had not detected any other leadership bid during the previous time slots. If at the beginning of the next time slot, any of the agents detect an average error in their observers during the last T seconds they

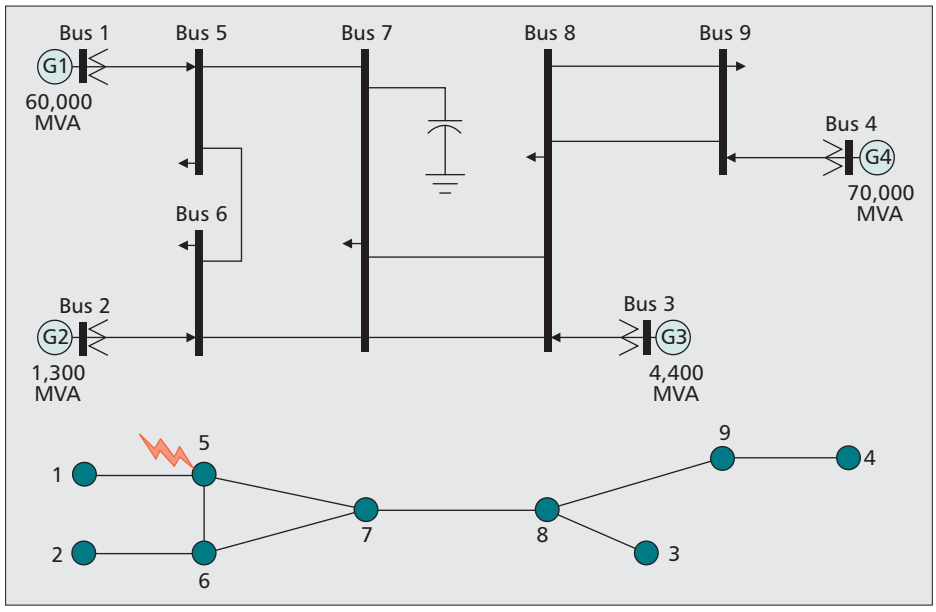

Figure 4. *A 9 bus power network and its graph representation.*

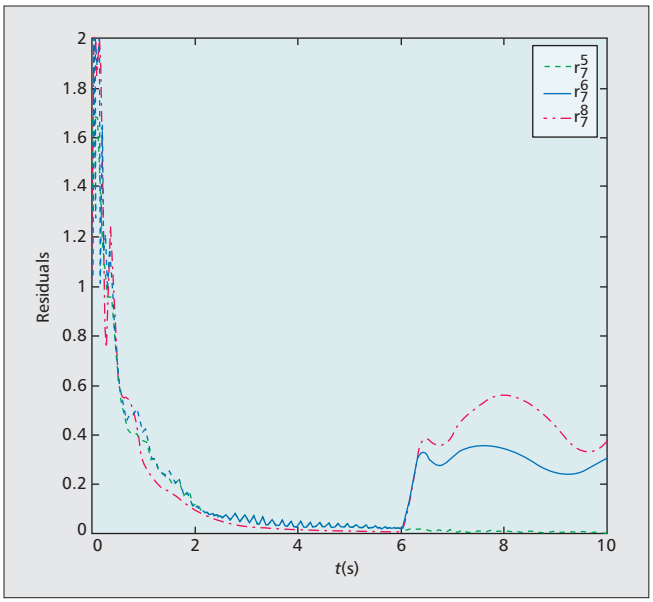

Figure 5. *The residuals generated at agent (node) 7. Since, r_7^6 and r_7^8 are much greater than zero and r_7^5 is approximately equal to zero, one can deduce that agent 5 is faulty.*

stop bidding for leadership from that time on. However if the error is sensed at the end of a time slot in which the agent was bidding for leadership, this agent stops the bidding for a random period of time ($\bar{\tau}_i$ is modified to reflect this idle time through choosing a new $\tilde{\tau}_i$), or "waits," and if it does not detect any other agent bidding for leadership in the network until the time is elapsed then it starts bidding for leadership again. Otherwise it will stop any future leadership bids as before. This process continues until an agent remains as the only agent capable of bidding for leadership. This procedure is described in Algorithm 2.

Remark 2: One might implement the algorithm letting only a subset of the agents bid for leadership, rather than all of them. A possible scenario that this may be desirable is; to replace a dysfunctional leader and letting only the neighbors of the previous leader to bid for leadership.

Comparison with Slotted ALOHA

In this section we compare this algorithm to an algorithm used commonly in data networks, namely `Slotted ALOHA`. Slotted Aloha is used to provide a packet collision avoidance mechanism in data transmission, and it is simple to describe. The idea is that each station in the network sends a data packet in the beginning of prescribed time frames, called time slots, to another station in the network. If the transmitted packet collides with other data packets in the network, then the station that has sent the packet waits for a randomly generated number of time slots and retransmits again when this waiting time has elapsed [15].

One can make an analogy between the algorithm introduced here and the `Slotted ALOHA`. If the bidding action is considered to be a data packet, taking this action is considered transmission, and the concurrent bidding action by different agents is considered collision, the algorithm proposed here aims to make sure that one and only one packet is delivered without collision. However, there are some important differences between `Slotted ALOHA` and the distributed leader selection algorithm proposed here. The most important difference is that in the leader selection algorithm if at any given time a leader is selected (the "packet" is delivered by any of the agents); the rest of the agents stop bidding for leadership (the rest would stop "transmission") indefinitely.

```
Wait for a random number of time slots τ̃_i ∈ {1, ···, n̄}
i is not the leader;
do Bid for Leadership loop
if It is the start of a time slot. then
    if any agent has taken a bidding action before i and
    waiting time for i is up then
      agent i will not be able to bid for leadership; return ;
    else
      agent i starts the bidding action;
    end if
    if any other agent is taking a bidding action at the same
    time as i then
      Stop taking bidding action; Wait for a random number
      of time slots;
    end if
    if It has been one time slot since the bidding action of
    i and no other agent has taken a bidding action. then
      i is the leader; return ;
    end if
end if
end loop
```

Algorithm 2. *Distributed leader selection.*

We present an example where Algorithm 2 is applied to address the problem of distributed leader selection as a conclusion to this section. Consider a network consisting of $N = 10$ agents, $\bar{n} = 5$, and let $T = 2$ s. The process of leader selection is depicted in Fig. 6.

Conclusions

In this article we argued that while a misbehavior in a network of interconnected systems does not necessarily result in a catastrophe, it is important to monitor agents in a network to detect if they behave in the way that they are expected to behave. We outlined a method to detect such misbehaviors in a distributed fashion, and we illustrated the application of the method to a network of interconnected electric machines, where each machine monitors those other ones that are directly connected to it.

Then we contemplated the possibility of accomplishing certain distributed tasks via making the agent to exhibit temporary misbehaviors. Specifically, we proposed an algorithm for distributed leader selection in a network of interconnected agents.

For future research directions the authors recommend two approaches that can be investigated simultaneously:

- Developing methods for fault detection and isolation/localization in distributed networked systems
- Critical misbehavior (failure) prevention and design robustness in networked systems through designing methodologies that are inherently robust to misbehavior or incorporating internal fail-safes that shut down an agent as it detects it has started exhibiting the first signs of misbehavior.

Acknowledgements

This work was supported in part by the European Commission through the VIKING project, the Swedish Research Council, the Swedish Foundation for Strategic Research, and the Knut and Alice Wallenberg Foundation.

References

[1] S. X. Ding, *Model-based Fault Diagnosis Techniques: Design Schemes*, Springer Verlag, 2008.
[2] S. X. Ding *et al.*, "Advanced Design Scheme for Fault Tolerant Distributed Networked Control Systems," *Proc. 17th IFAC World Congress*, Seoul, Korea, July 2008, pp. 569–74.
[3] W. H. Chung, J. L. Speyer, and R. H. Chen, "A Decentralized Fault Detection Filter," *J. Dynamic Systems, Measurement, and Control*, vol. 123, no. 2, 2001, available: http://link.aip.org/link/?JDS/123/237/1, pp. 237–47

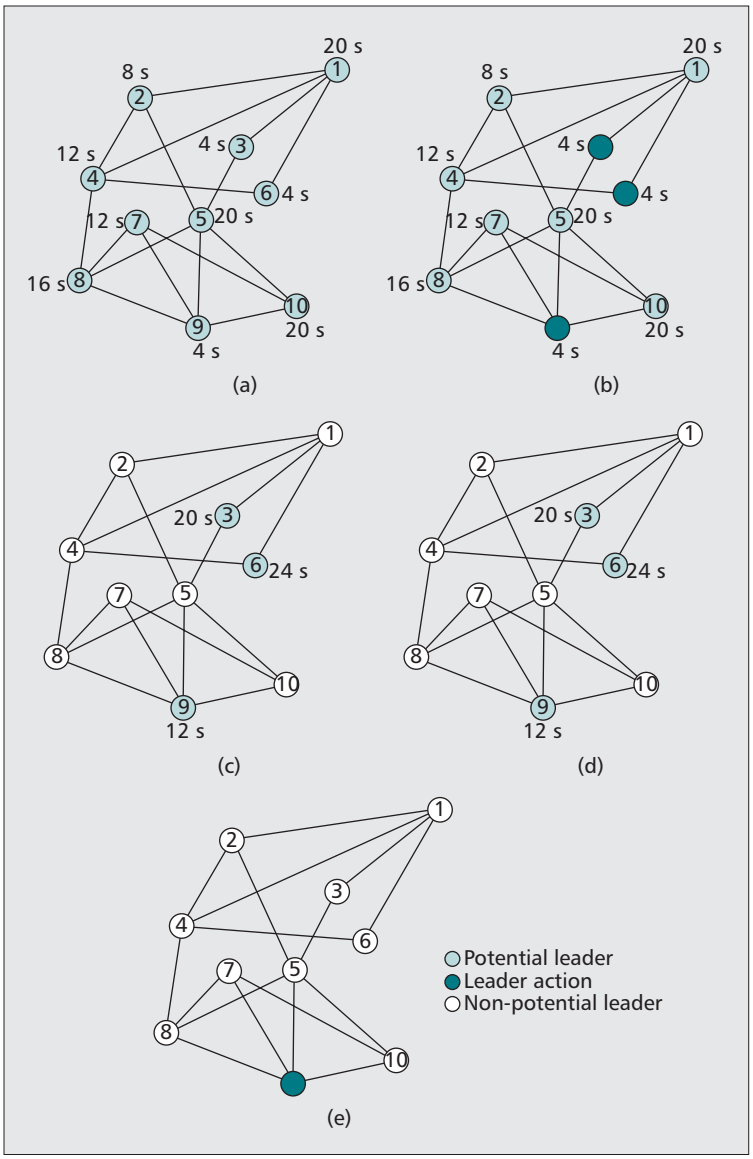

Figure 6. *The underlying graph of the network. The numbers on the circles denote the agents labels, and the smaller ones outside the circles are the time at which the agents will take their corresponding bidding action.*

[4] F. Pasqualetti, A. Bicchi, and F. Bullo, "Consensus Computation in Unreliable Networks: A System Theoretic Approach," *IEEE Trans. Automatic Control*, 2010, to appear, available online at http://www.fabiopas.it/papers/FP-AB-FB-10a.pdf.

[5] I. Shames et al., "Distributed Fault Detection for Interconnected Second-Order Systems," *Automatica*, vol. 47, no. 12, 2011, pp. 2757–64.

[6] E. Handschin *et al.*, "Bad Data Analysis for Power System State Estimation," *IEEE Trans. Power Apparatus and Systems*, vol. 94, no. 2, 1975, pp. 329–37.

[7] E. Scholtz and B. Lesieutre, "Graphical Observer Design Suitable for Large-Scale DAE Power Systems," *Proc. IEEE Conf. Decision and Control*, Cancun, Dec. 2008, pp. 2955–60.

[8] V. Gupta, C. Langbort, and R. M. Murray, "On the Robustness of Distributed Algorithms," *Proc. 45th IEEE Conf. Decision and Control*, 2006, pp. 3473–78.

[9] I. D. Couzin *et al.*, "Effective Leadership and Decision-Making in Animal Groups on the Move," *Nature*, vol. 433, no. 7025, 2005, pp. 513–16.

[10] L. D. Mech, "Leadership in Wolf, Canis Lupus, Packs," *Canadian Field-Naturalist*, vol. 114, no. 2, 2000, pp. 259–63.

[11] W. Wang and J. J. E. Slotine, "A Theoretical Study of Different Leader Roles in Networks," *IEEE Trans. Automatic Control*, vol. 51, no. 7, 2006, pp. 1156–61.

[12] F. Bullo, J. Corts, and S. Martnez, Distributed Control of Robotic Networks, *Series in Applied Mathematics*, Princeton, 2009.

[13] A. J. King, D. D. Johnson, and M. V. Vugt, "The Origins and Evolution of Leadership," *Current Biology*, vol. 19, no. 19, 2009, pp. R911–R916.

[14] I. Shames *et al.*, "Distributed Leader Selection Without Direct Inter-Agent Communication," *2nd IFAC Wksp. Estimation and Control of Networked Systems*, Annecy, France, 2010, pp. 221–26.

[15] S. Ghez, S. Verdu, and S. C. Schwartz, "Stability Properties of Slotted Aloha with Multipacket Reception Capability," *IEEE Trans. Automatic Control*, vol. 33, no. 7, 1988, pp. 640–49.

Biographies

IMAN SHAMES (imansh@kth.se) is currently a Postdoctoral Researcher at the ACCESS Linnaeus Centre, the Royal Institute of Technology (KTH), Stockholm, Sweden. He received his B.S. degree in Electrical Engineering from Shiraz University, Iran in 2006, and the Ph.D. degree in engineering and computer science from the Australian National University, Canberra, Australia. He has been a visiting researcher at the Royal Institute of Technology (KTH) in 2010, the University of Tokyo in 2008, and at the University of Newcastle in 2005. His current research interests include multi-agent systems, sensor networks, systems biology, and distributed fault detection and isolation.

ANDRÉ TEIXEIRA (andretei@kth.se) received his Master's degree in Electrical and Computers Engineering in 2009 from the Faculdade de Engenharia da Universidade do Porto, Portugal. Currently he is a Ph.D. student at the department of Automatic Control at KTH - Royal Institute of Technology, Sweden. His main research interests are secure control, fault detection and isolation, power systems, and multi-agent systems.

HENRIK SANDBERG (hsan@kth.se) received the M.Sc. degree in Engineering Physics in 1999 and the Ph.D. degree in Automatic Control in 2004, both from the Lund University, Sweden. In 2005–2007, he was a postdoctoral scholar at the California Institute of Technology in Pasadena, USA. Since 2008, he is an Assistant Professor in the Automatic Control Laboratory at the KTH Royal Institute of Technology in Stockholm, Sweden. He has also held visiting appointments at the Australian National University and the University of Melbourne, Australia. His research interests include secure networked control, power systems, model reduction, and fundamental limitations in control. He was the winner of the Best Student-Paper Award at the IEEE Conference on Decision and Control in 2004, and is an associate editor of Automatica

KARL H. JOHANSSON (kallej@kth.se) is Director of the ACCESS Linnaeus Centre and Professor at the School of Electrical Engineering, Royal Institute of Technology, Sweden. He is a Wallenberg Scholar and has held a Senior Researcher Position with the Swedish Research Council. He received M.Sc. and Ph.D. degrees in Electrical Engineering from Lund University in Sweden. He has held visiting positions at UC Berkeley (1998–2000) and California Institute of Technology (2006–07). His research interests are in networked control systems, hybrid and embedded control, and control applications in automation, communication, and transportation systems. He was a member of the IEEE Control Systems Society Board of Governors 2009 and the Chair of the IFAC Technical Committee on Networked Systems 2008–2011. He has been on the Editorial Boards of IEEE Transactions on Automatic Control and the IFAC journal Automatica, and is currently on the Editorial Boards of IET Control Theory and Applications and the International Journal of Robust and Nonlinear Control. He was the General Chair of the ACM/IEEE Cyber-Physical Systems Week (CPSWEEK) 2010 in Stockholm. He has served on the Executive Committees of several European and national research projects in the area of networked embedded systems. He was awarded an Individual Grant for the Advancement of Research Leaders from the Swedish Foundation for Strategic Research in 2005. He received the triennial Young Author Prize from IFAC in 1996 and the Peccei Award from the International Institute of System Analysis, Austria, in 1993. He received Young Researcher Awards from Scania in 1996 and from Ericsson in 1998 and 1999.'