An energy-efficient and self-triggered control method for robot swarm networking systems

Heejung Byun and Soo-mi Yang

Abstract
Robot swarm networking systems (RSNS) exhibit complex emergent behavior by using local control laws based on spatial information from nearby environment and adjacent robot agents. The consensus behavior of the RSNS depends on a set of parameters of robot agent algorithms or system parameters for their operation, issued mainly by the operator. The challenge in the RSNS is developing techniques for the operator to interact with the RSNS in order to make system behavior adaptive to changes in system configuration and for operator commands without having to handle them individually. Another challenge is saving energy consumption over the robot agents in the RSNS by reducing the number of information exchange between robot agents when the system configuration is spread out the network from the operator. To address these issues, this paper presents an energy-efficient control approach for system configuration propagation with self-triggering control. The proposed method controls the RSNS operation by indirectly propagating the system configuration within the framework of local rules. Moreover, a self-triggered propagation model is designed according to the convergence rate of configuration propagation in order to save and to balance energy consumption among robot agents in the RSNS. This model is then extended to an optimal timing control, where the operator determines its next input time without having to keep track of all the states of robot agents. Theoretical analysis and simulation results are performed to demonstrate the superiority of the proposed method.

Keywords
Robot swarm networking system, energy-efficiency, self-triggered control

Introduction
A robot swarm networking system (RSNS) consists of a large number of robot agents with local sensing and communication capabilities while maintaining decentralized control based on underlying laws. Swarm robotics systems are biologically inspired by nature systems in which large numbers of simple agents perform complex collective behaviors automatically through local interactions between themselves. Given that the consensus behavior generated is collective, as well as robust to failure of an individual robot agent, RSNS is useful in complex tasks including environmental exploration, large-scale search and rescue, and protection. In an RSNS, an operator commands robot agents to carry out mission goals or tasks. A linear matrix inequality (LMI)-based design method was for multi-agent systems with leader-follower structures. The resulting behaviors the RSNS generates depend on a set of parameters of robot agent algorithms or system parameters for their operation. The types of control an operator can exert on the robot agents are the following: switching between algorithms that implement desired consensus behavior, changing parameters of algorithms, controlling through selected robot agent members, remote programming, new software downloading, and reprogramming.

As such, in order to perform supervisory control of consensus behaviors, one main challenge is to design of systems for the operator to convey appropriate parameter adjustments or system configuration independent of the number of robot agents as intended goals change. Specifically, when the user input is sequentially applied to the RSNS, the operator needs to estimate the optimal time to allot for the next input to the

Department of Information Communication Technology, The University of Suwon, Hwaseong, Republic of Korea

Corresponding author:
Soo-mi Yang, Department of Information Communication Technology, Suwon University, 17 Wauan-gil, Hwaseong, Gyeonggi-do, Republic of Korea.
Email: smyang@suwon.ac.kr

Creative Commons CC BY: This article is distributed under the terms of the Creative Commons Attribution 4.0 License (https://creativecommons.org/licenses/by/4.0/) which permits any use, reproduction and distribution of the work without further permission provided the original work is attributed as specified on the SAGE and Open Access pages (https://us.sagepub.com/en-us/nam/open-access-at-sage).
system. This is because system performance is affected by the time between control inputs that the operator applies to the system. However, operators have difficulty understanding the evolution of the system. Another challenge is to design of systems for saving the energy consumption over the robot agents in RSNS. The robot agent members in the RSNS are generally battery-powered, so they can be easily depleted of energy if they remain active while these controls take place in the RSNS. Consequently, energy will be imbalanced among robot agents and RSNS will have shorter lifetime. Furthermore, the operator is not aware of these local battery states of robot agents and do not identify the number of robot agents the system configuration has been propagated to. Thus, energy is wasted because of the continuous spread of such controls to the system.

Walker et al. focused on two methods of information propagation (flooding and consensus methods) and compared the ability of operators to manage the multi-agent system to the desired goals. In the flooding method, each robot agent explicitly matches the value of user command. Meanwhile, in the consensus method, each robot agent matches the average value of user command of all the neighbors it senses. They also investigated the use of dynamically selected leaders that are directly controlled by the operator to guide the rest of the systems. Goodrich et al. worked on a leader-based control of systems using tele-operated leaders based on Couzin’s control laws. Pendleton and Goodrich similarly implemented a leader-based model using both virtual robot agents and an operator as leaders in a system. McLurkin et al. proposed a single-hop broadcast algorithm for downloading new software, but agents that are too far from the user will not be reprogrammed. Li et al. proposed an architecture for multi-robot agent communication networks, in which agents are clustered to one or multiple systems and each system can be monitored by some central servers through a wireless mesh backbone. Chen et al. proposed a generic framework for the multi-agent planning solution, that is the determination of the number of agents. Dmarogonas et al. proposed event-driven strategies to reduce the number of the control updates. Each agent computes its next update time and performs a self-triggered setup. The aforementioned existing studies addressed the interaction problems between the operator and the multi-agent system, but limited work has focused on how system configuration should be spread through the RSNS via operator-agent interactions while achieving the system’s energy efficiency. We previously proposed a simple model of configuration propagation by a human operator. Such model allowed each agent to automatically drive the system parameters to the desired configuration without considering the system’s energy efficiency.

This paper extends the previous work and proposes a dual control approach for indirect propagation of system configuration with an energy efficient self-triggering control in RSNS. First, we propose a method that influences RSNS operation by indirectly propagating system configuration from the operator within the framework of local rules in the RSNS. Second, we design a self-triggering propagation model of robot agent state, in which each robot agent autonomously determines when to send its configuration state update to the neighbors depending on the configuration propagation rate. Then, we extend the self-triggering model to an optimal timing control, where the operator computes the optimal time to give sequential control input to the RSNS. Finally, based on theoretical analysis, we provide insights into the performance of the proposed method by deriving the convergence to the desired goal and the stability of the proposed system.

Our primary contributions are summarized as follows: (a) The new model of self-triggered propagation based on the agreement control laws is proposed. From the proposed method, robot agents autonomously determine when to send the state update to their neighboring agents. (b) The new model of optimal timing control of a sequence input is defined, which helps the operator compute an optimal time to send sequential command to the RSNS. (c) A theoretical analysis for achieving desired system performance while saving energy consumption is provided. From the analysis, we derive the condition of system parameters for the system to be stable.

The rest of the paper is organized as follows. In Section II, we present our proposed method which includes design of configuration state and self-triggered propagation model. In addition, we provide a stability analysis of the proposed method, which shows the system stability with bounded error. In Section III, we discuss the simulation results where we compare our proposed method with other existing methods to test the effectiveness of the method in terms of energy consumption and convergence. Finally, Section IV concludes the paper with remarks for future work.

Proposed method

Configuration state control model

We consider an RSNS consisting of \( N \) robot agents and an operator. Let \( \mathbf{N} = \{1, 2, \ldots, N\} \) denote the set of robot agents in the RSNS. The set of neighboring robot agents of agent \( i \) (\( 1 \leq i \leq N \)) is denoted as \( \mathbf{N}_i (\subseteq \mathbf{N}) \), and its cardinality is \( N_i \). The \( N \)-th robot agent in \( \mathbf{N} \) is referred to as a gateway agent. In this section, a system configuration propagation model is proposed. In the model, the operator controls the RSNS by conveying configuration setting to the single gateway agent, then influences indirect propagation to the rest of the RSNS based on autonomous control laws each robot agent uses. No control can be exerted over individual agent members, only over the RSNS as a whole.

For the configuration state control, a discrete-time formula is developed in which the control updates take place...
every control period $\tau_s$. Thus, the time is divided into 
$[t, t+1)$, $t=0,1,\ldots$, with time duration equal to $\tau_s$. 
First, $u(t) \in \mathbb{R}^n$ is denoted as the desired system 
configuration state vector issued by the operator at time slot $t$, 
such as command inputs, parameter setting, and parameter 
changes that the operator wants to achieve with the RSNS. 
Let the vector of the configuration states of all robot agents 
at time slot $t$ as $x(t) = [x_1(t), x_2(t), \ldots, x_N(t)]^T$, where $x_i(t) = [x_i^1(t), x_i^2(t), \ldots, x_i^n(t)]^T \in \mathbb{R}^n$ is the configuration 
state vector of robot agent $i$ at time slot $t$ and individual 
agent has $n$-dimensional configuration state space. 

The operator interacts with the RSNS by applying 
the desired configuration input to the gateway agent, 
while the other agents control their respective 
configuration state vectors and propagate them by 
interacting with each other. A simple configuration 
model has been introduced in the previous work.\(^{30}\) 
Specifically, the previous work used the following local 
law of robot agent $i$ ($i \in \mathbb{N}\setminus N$) in the RSNS:

$$
\dot{x}_i(t+1) = x_i(t) + \alpha \sum_{j \in \mathbb{N}\setminus N} (x_j(t) - x_i(t)) + \alpha 1_{(i,N)}(x_N(t) - x_i(t)),
$$

(1)

where, $\alpha$ is the parameter that determines the convergence 
speed and $1_{(i,N)}$ is the indication function: the value of $1_{(i,N)} = 1$, if node $N$ (the gateway) is a neighbor of node $i$ and $1_{(i,N)} = 0$, otherwise. The operator gives commands or informs the gateway agent of the desired configuration settings and parameter changes with $u(t)$, after which the user input is transformed into configuration state vector and propagated to the RSNS. 

To achieve this, the user input is first applied for the single gateway agent by letting $x_N(t) = u(t)$, then the interactions between the remaining agent members are handled autonomously based on equation (1). 

Specifically, each agent in the RSNS interacts with its neighbors with configuration state vector $x_i$ rather than actual values of system configuration $u$. That is, the user's intended command is conveyed only to the gateway agent and each of the remaining agents performs indirect parameter setting aided by the proposed dynamics instead directly propagating the user command. In this way, the operator takes actions independent of the number of agents and supervises the RSNS as a single entity, hence a control complexity of $O(1)$.

**Self-triggered propagation model**

When the system configuration issued by the operator 
is propagated throughout the RSNS, an important aspect to consider is to save energy consumption by reducing the number of messages to be exchanged while keeping the energy consumption of robot agent members balanced. In this section, a self-triggered propagation model is designed for each robot agent based on the configuration states. The self-triggering state of robot agent $i$ ($i \in \mathbb{N}$) is denoted as $r_i$, which is determined by the following rule:

$$
\begin{align*}
    r_i(t+1) &= r_i(t) - \epsilon_i (r_i(t) - r_{\text{min}}) \\
    &+ \left\| \sum_{j \in \mathbb{N}\setminus N} (x_j(t) - x_i(t)) \right\|_2 + \left\| \sum_{j \in \mathbb{N}\setminus N} x_j(t) \right\|_2, \\
\end{align*}
$$

(2)

where $\epsilon_i$ is the control parameter to be chosen within the range of $(0, 2)$, and $r_{\text{min}}$ is the minimum message triggering probability that can be set by the operator. 

When robot agent $i$ calculates $r_i$, it independently generates a random value $\omega$ following the uniform distribution within [0, 1]. If the value of $r_i$ is greater than $\omega$, the robot agent triggers transmission of its configuration state update to the neighbors. The higher the value of $r_i$, the higher the probability of message transmission. According to (2), the self-triggered propagation model works in such a way that the minimum message triggering probability is achieved when robot agent $i$ reaches an equilibrium point, where the state of the agent’s configuration converges to the user command $u$. As the configuration state of robot agent $i$ is closer to the averaged state among its neighbors, the value of $r_i$ goes near to $r_{\text{min}}$. The configuration state of an agent becomes identical to those of its neighbors, and the value of $r_i$ decreases; hence, the lower the probability of message exchanges and energy savings. On the contrary, as the configuration state of agent $i$ is different from those of surrounding neighbors, the value of $r_i$ increases, resulting in a higher message triggering probability and frequent packet transmission. This accelerates the convergence speed of the configuration state values to $u$ for all agents in the RSNS.

**Feasibility analysis**

In this section, we will analyze the feasibility of the proposed RSNS model for an operator to achieve desired system properties of the RSNS, such as system convergence and stability. We denote $x_i$ as a steady state vector of $x(t)$, which is the state where the proposed RSNS under the local control law can asymptotically converge. Specifically, we denote $x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T$, where $x_{it} \in \mathbb{R}^n$ is the steady state of $x_i(t)$. Also, we denote $r_i = [r_{i1}, r_{i2}, \ldots, r_{in}]^T$, where $r_{it}$ is the steady state of $r_i(t)$.

For the purpose of simplicity, we consider that the sequence of system configuration set $u(t)$ is intermittently changed by the operator and and the value of input $u$ is piece-wise constant while the same system configuration is maintained such that $u(t) \approx u$. By letting $x_i(t+1) = x_i(t)$ in equation (1), we obtain the following steady states for all $i \in \mathbb{N}\setminus N$ as
where

\[ x_{i\Delta} = \frac{1}{N_{i\Delta}} \sum_{j \in N_{i\Delta}, j \neq i} x_{j\Delta} \]

\[ = \frac{1}{N_i} (N_{i,1} x_{1\Delta} + N_{i,2} x_{2\Delta} + \cdots + N_{i,N} x_{N\Delta}), \]  

(3)

Then, equation (3) is rewritten as

\[ \mathbf{A} \mathbf{x}_i = \mathbf{0}, \quad (4) \]

where

\[
\mathbf{A} = \begin{bmatrix}
\frac{1}{N_1} & \frac{1}{N_{1,2}} & \cdots & \frac{1}{N_{1,N}} \\
\frac{1}{N_{2,1}} & \frac{1}{N_2} & \cdots & \frac{1}{N_{2,N}} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{1}{N_{N,1}} & \frac{1}{N_{N,2}} & \cdots & 1
\end{bmatrix}.
\]

Let \( \mathbf{L} \) is a random walk normalized Laplacian matrix, defined as \( \mathbf{L} = \mathbf{D} - \mathbf{S} \) where \( \mathbf{D} \) is the diagonal matrix and

\[ S_{ij} = \begin{cases} 
\frac{1}{N_i} & \text{if } i = j = 1 \\
0 & \text{otherwise}.
\end{cases} \]

Therefore, the elements of \( \mathbf{L} \) are given by

\[ L_{ij} = \begin{cases} 
1 & \text{if } i = j \text{ or } i = j = 1 \\
\frac{1}{N_i} & \text{if } 1(2, j) \text{ or } 1(1, 2) \\
0 & \text{otherwise}.
\end{cases} \]

(6)

Consider a vector \( \mathbf{v} = [x_1, x_2, \ldots, x_N]^T \), then we will have the action of \( \mathbf{L} \) as

\[ [\mathbf{L} \mathbf{v}]_i = x_i - \frac{1}{N_{i\Delta}} \sum_{j \in N_{i\Delta}, j \neq i} x_j. \]

(7)

We observe that \( \mathbf{I} = [1, \ldots, 1]^T \) is an eigenvector of \( \mathbf{L} \) with eigenvalue 0, since for this vector \( x_i \) always equals the average of its neighbors’ values. As in \( \mathbf{L}, \mathbf{A} \) in equation (4) is a random walk normalized Laplacian matrix such as the constant \( x_i \) are eigenvectors of eigenvalue 0. In the proposed algorithm, the user input is directly applied for the \( N \)-th robot agent, and then we have

\[ x_{N\Delta} = u. \]

(8)

Accordingly, we redefine \( \mathbf{x}_i = [x_{1\Delta}, x_{2\Delta}, \ldots, x_{(N-1)\Delta}]^T \) and replace \( x_{N\Delta} \) by \( u \). Then, we reduce the order of equation (4) by the following equation:

\[ \mathbf{A} \mathbf{x}_i = \mathbf{b}, \quad (9) \]

where

\[
\mathbf{A} = \begin{bmatrix}
1 & -\frac{1}{N_1} & \cdots & -\frac{1}{N_{1,N-1}} \\
-\frac{1}{N_2} & 1 & \cdots & -\frac{1}{N_{2,N-1}} \\
\vdots & \vdots & \ddots & \vdots \\
-\frac{1}{N_{N-1}} & -\frac{1}{N_{N-2}} & \cdots & 1
\end{bmatrix}.
\]

\[
\mathbf{b} = \begin{bmatrix}
\frac{1}{N_1} (1(1,N) u) \\
\frac{1}{N_2} (1(2,N) u) \\
\vdots \\
\frac{1}{N_{N-1}} (1(N-1,N) u)
\end{bmatrix}^T.
\]

In order to get \( x_{i\Delta} \), we divide into two cases: \( 1(1,N) = 1 \) and \( 1(1,N) = 0 \). In the case of \( 1(1,N) = 1 \), obviously \( x_{N\Delta} = u \), and we obtain the following:

\[ N_{N} x_{N\Delta} = \sum_{j \in N_{i\Delta}, j \neq N} x_{j\Delta} + u = (N_N - 1)x_{\Delta} + u. \]

(10)

This shows that \( x_{i\Delta} = u \) for all \( i \). In the case of \( 1(1,N) = 0 \), the value of \( x_{i\Delta} \) converges to the average of its neighbors’ state values. If there exists a multi-hop path from robot agent \( i \) to any robot agent \( k \in N_N \), we obtain \( x_{i\Delta} = u \) according to equations (9) and (10). If there is no such multi-hop path from robot agent \( i \) to any robot agent \( k \in N_N \), it means that robot agent \( i \) is isolated and disconnected from the network. Except for these extreme cases, we get the following conclusion:

\[ x_{i\Delta} = [x_{1\Delta}, x_{2\Delta}, \ldots, x_{N\Delta}]^T = [u, u, \ldots, u]^T. \]

(11)

Next, based on the results of equations (2) and (11), we derive \( \mathbf{r}_i \) as follows:

\[ \mathbf{r}_i = [r_{1\Delta}, r_{2\Delta}, \ldots, r_{N\Delta}]^T = [r_{\min}, r_{\min}, \ldots, r_{\min}]^T. \]

(12)

Equations (11) and (12) show that the configuration states of all robot agents converge to the desired system configuration so that the operator can successfully achieve the indirect system configuration under the proposed controller. Also, as the configuration states of the robot agents converge to the desired configuration, the activation probabilities of all robot agents converge equally to \( r_{\min} \).

Next, we need to prove that the proposed RSNS model ensures stability. When the system is asymptotically stable, the trajectory will converge to the steady state derived from equations (11) and (12) as time goes to infinity. Let \( \delta x_i = x_i - x_{0i} \) and \( \delta r_i = r_i - r_{\min}, \forall i \in N \). Then, equations (1) and (2) are rewritten as
where each 
\[ \det(A) = \alpha \sum_{j \in N_1} (\delta x_j(t) - \delta x_i(t)) \]

and 
\[ f_r(\delta x(t), \delta r(t)) = -e_r \delta r(t) \] (14)

We approximate the nonlinear system as described in
applications (13) and (14) via linearization. Then, the
first-order linear approximation is derived as follows:
\[ \delta x_i(t + 1) = \delta x_i(t) + \alpha \sum_{j \in N_1} (\delta x_j(t) - \delta x_i(t)) \]
\[ \delta r_i(t + 1) = (1 - e_r) \delta r_i(t) \] (15)

We define \( e(t) = [\delta x^T(t) \quad \delta r^T(t)]^T \)
where
\[ \delta x(t) = [\delta x_1(t) \quad \delta x_2(t) \cdots \delta x_{N-1}(t)]^T \]
\[ \delta r(t) = [\delta r_1(t) \quad \delta r_2(t) \cdots \delta r_{N-1}(t)]^T \] (16)

Then, the state space model for the RSNS is written as

\[ e(t + 1) = A_e e(t) \] (17)

where
\[ A_e = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \]
\[ A_{11} = \begin{bmatrix} 1 - \alpha N_1 & \alpha 1_{(1,2)} & \cdots & \alpha 1_{(1,N-1)} \\ \alpha 1_{(2,1)} & 1 - \alpha N_2 & \cdots & \alpha 1_{(2,N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha 1_{(N-1,1)} & \alpha 1_{(N-1,2)} & \cdots & 1 - \alpha N_{N-1} \end{bmatrix} \]
\[ A_{12} = A_{21} = 0 \in \mathbb{R}^{(N-1) \times (N-1)}, \]
\[ A_{22} = \text{diag}(1 - e_r) \in \mathbb{R}^{(N-1) \times (N-1)}. \] (18)

Note that the matrix \( A_e \) is a block diagonal matrix
where each \( A_{ii} \) is square and all entries above/below the
\( A_{ii} \) blocks are zero. Then, the characteristic polynomial
det \( (A_e - \lambda I) \) is determined as:
\[ \prod_{i=1}^{2} \det(A_{ii} - \lambda I). \] (19)

Therefore, the set \( \lambda(A_e) \) of eigenvalues of \( A_e \) is the
union of the set of eigenvalues of the diagonal blocks
\( A_{ii} \) such that
\[ \lambda(A_e) = \lambda(A_{11}) \bigcup \lambda(A_{22}). \] (20)

Since \( e_r \) is selected within the range of \((0, 2)\), every
eigenvalue of matrix \( A_{22} \) obviously lies within the unit
circle.

For the case of matrix \( A_{22} \), its every eigenvalue obviously
lies within the unit circle. Considering of \( A_{11} \), it
takes the form of a symmetric Laplacian matrix \( L \)
such that
\[ A_{11} = I - \alpha \tilde{L} \] (21)

where \( \tilde{L} = \tilde{D} - \tilde{S}, \tilde{D} \) is diagonal matrix whose \( i \)-th diago-
nal entry is \( N_i \) and \( \tilde{S} \) is given by
\[ \tilde{S}_i = \begin{cases} 1 & \text{if } 1_{(i,j)} = 1 \\ 0 & \text{otherwise.} \end{cases} \] (22)

Therefore, the elements of \( \tilde{L} \) is given by
\[ \tilde{L}_{ij} = \begin{cases} N_i & \text{if } i = j \\ -1 & \text{if } 1_{(i,j)} = 1 \\ 0 & \text{otherwise.} \end{cases} \] (23)

Note that \( \sum_{i} \tilde{L}_{ii} = \sum_{j} \tilde{L}_{jj} = 0 \) and \( \tilde{D}_{ii} = \sum_{j \in N_i, j \neq i} 1_{(i,j)} \).

Let \( z = [z_1 \quad z_2 \cdots z_{N-1}]^T \). Then, for all \( z \in \mathbb{R}^{N-1}, \)
\[ z^T \tilde{L} z = \sum_{(i,j)} 1_{(i,j)} (z_i - z_j)^2 \] (24)

where \( \sum_{(i,j)} \) is the sum of all pairs \((i,j)\) with \( 1_{(i,j)} = 1 \).

For eigenvector set \( v \) of eigenvalue set \( \lambda \), this tells us
that
\[ v^T \tilde{L} v = \lambda v^T v \]
\[ = \sum_{(i,j)} 1_{(i,j)} (v_i - v_j)^2 \geq 0. \] (25)

If all entries of \( z \) are the same, then \( z^T \tilde{L} z = 0 \), so the
constant vector are eigenvectors of eigenvalue 0. To
determine the upper bound of eigenvalues, we use the
random walk normalized Laplacian introduced in equation
(6), which is redefined as
\[ L = \tilde{D}^{-1} \tilde{L} \] (26)

From the definition of \( \tilde{L} \) and equation (26), we can
derive the following:
\[
\mathbf{L} = \hat{\mathbf{D}}^{-1}(\hat{\mathbf{D}} - \mathbf{\hat{S}}) \\
= 1 - \hat{\mathbf{D}}^{-0.5}(1 - \hat{\mathbf{L}}_{\text{sym}})\hat{\mathbf{D}}^{-0.5},
\]

where \( \hat{\mathbf{L}}_{\text{sym}} \) is the symmetric normalized matrix of \( \hat{\mathbf{L}} \), such as \( \hat{\mathbf{L}}_{\text{sym}} = \hat{\mathbf{D}}^{-0.5}\hat{\mathbf{L}}\hat{\mathbf{D}}^{-0.5} \), whose eigenvalues of any graph lie between 0 and 2. Since its eigenvalues of \( \mathbf{L} \) agree with those of \( \hat{\mathbf{L}}_{\text{sym}} \), and \( \mathbf{L} = \hat{\mathbf{D}}\mathbf{L} \), we drive the upper bounds of eigenvalues of \( \mathbf{L} \), such that \( 0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{N-1} \leq 2N_{\text{max}} \), where \( N_{\text{max}} = \max_{i}N_i \).

In order for the proposed controller to be stable, the characteristic polynomial of \( \mathbf{A}_1 = \mathbf{I} - \alpha \hat{\mathbf{L}} \) should have all zeros within the unit circle. Accordingly, the proposed system is asymptotically stable if the control parameters satisfy the following relationships:

\[
0 < \alpha < \frac{1}{N_{\text{max}}}. \tag{28}
\]

With the condition of equation (28), we can claim that the proposed system is stable and the error dynamics converges to zero exponentially. This result guarantees the autonomous adaptation of robot agent’s configuration and energy management as well according to user-defined system configuration. It also shows feasibility for an operator to use the proposed RSNS model to automatically propagate system configuration adjustments to the RSNS while guaranteeing energy efficiency.

**Optimal timing control of sequential input**

Another important function of the RSNS is to estimate the system state so the operator can change or properly give sequential control input to the RSNS. To support this feature, we present an optimal timing control for sequential input. Based on the self-triggering state control in equation (2), the gateway robot agent determines the optimal time to give the next input based on the value of \( r_N \) as follows:

\[
r_N(t+1) = r_i(t) - \epsilon_e(r_N(t) - r_{\text{min}}) \\
+ \frac{\sum_{j \in N_0} (x_j(t) - x_N(t))}{\sum_{j \in N_0} x_j(t)}. \tag{29}
\]

The gateway robot agent provides a feedback of \( r_N \) value to the operator. The foregoing stability analysis indicates that when the states of all robot agents converge to the desired configuration, the system is then driven to \( r_N = r_{\text{min}} \). Therefore, the operator can understand system states and estimate the timing to give for the next control inputs, considering the feedback from the gateway robot agent only. Define \( r_d = r_{\text{min}} - r_N \). Then, the operator gives a new input to the RSNS when \( |r_d| < \epsilon \), where \( \epsilon \) is the tolerance for convergence.

**Simulation results**

**Configuration**

The simulation area is 100 m \( \times 100 \text{m} \), where the entire network is divided into equally shaped grids, and the robot agents are uniformly deployed. We set \( N = 50 \), and the robot agent members are arbitrary connected. The gateway robot agent is denoted as \( R_{90} \), which is chosen randomly by the operator. The channel capacity is set to 200 kbps, and the transmission range and carrier sense range to 20 and 40 m, respectively. The current consumption for \( T_s \), \( R_s \), and mode switch is set to 17.4, 19.7, and 10.05 mA, respectively. The mode switch time and backoff time are set to 300 \( \mu \text{s} \) and 30 ms. The parameters of the values in equation (2) are set to \( \epsilon_e = 1 \) and \( r_{\text{min}} = 0.1 \), respectively. We use the following metrics to observe the performance of the proposed scheme.

- Energy difference ratio: The difference in energy consumption between the robot agent with the highest energy consumption ratio \( (E_{\text{max}}) \) and that with lowest \( (E_{\text{min}}) \) in the RSNS. The energy consumption ratio of each robot agent is the ratio of the robot agent’s consumed power to the initial power \( E_0 \). Then, the energy difference ratio is evaluated as \( (E_{\text{max}} - E_{\text{min}}) \). The energy difference ratio approaching zero indicates energy balance.
- Residual energy ratio: The available energy of the robot agent with the highest energy consumption rate \( (E_{\text{max}}) \) in the RSNS, which is represented as \( (1 - E_{\text{min}}) \times 100 \% \). The higher residual energy ratio means the longer lifetime of the RSNS.
- Convergence time: The time to complete transmission of the entire command.

**Results and discussion**

**Configuration and self-triggering state.** Figure 1 shows the numerical results of the proposed controller. We consider four robot agents in a RSNS. The configuration and self-triggering states of three agents are denoted as \( x_1, x_2, x_3 \) and \( r_1, r_2, r_3 \), respectively. We set the initial values of \( x_1, x_2, x_3 \) differently. The user input is initially set to \( u = 4 \) and changed to \( u = 2 \) and 3 at 30 min, 60 min, respectively. The minimum activation probability value is set to \( r_{\text{min}} = 0.1 \). As shown in Figure 1(a), the configuration state of each robot agent changes according to the user input and are successfully converged to the user input. For the self-triggering states as shown in Figure 1(b), the values of \( r_1, r_2, r_3 \) converge to the pre-determined value of \( r_{\text{min}} \) at the steady state. When the user input changes at 30 min, 60 min, the self-triggering states of the agents are also adjusted and
finally converged to the value of $r_{\text{min}}$.

**Configuration state adaptation.** We examine the trajectories of system configuration state and operation state corresponding to control input changes during a runtime. Simulations are conducted over the interval from 0 min to 40 min in 6 sec increments. The control input $u = \frac{[2, 2]^T}{C_1 38} \in \mathbb{R}^2$ is first applied and then changed to $u = \frac{[4, 4]^T}{C_1 38}$ at 20 min. Randomly chosen are 3 out of 49 robot agents with time trajectories of their configuration states $x_i = [x^1_i, x^2_i]^T \in \mathbb{R}^2$ and operation states $r_i$ ($i = 1, 2, 3$). To show the performance of controlling configuration state of each robot agent, we set the initial state for each differently, such that the initial configuration states of robot agents 1, 2, and 3 are set to $\frac{[1, 1]^T}{C_1 38}$, $\frac{[3, 1]^T}{C_1 38}$, $\frac{[4, 4]^T}{C_1 38}$, respectively. In this simulation, an error condition is implemented so the state of $x_2$ is not updated during $[0.5, 2]$ min and $[20, 22]$ min because of a malfunction of robot agent 2. During the malfunction period, robot agent 2 stops updating $x_2$ and transmits only the last updated state value to its neighbor robot agents.

Figure 2 shows the configuration state adaptation behavior for the consensus and the proposed methods according to control input changes. Both methods show that the configuration state values of all robot agents are adjusted according to the user control input changes. However, in the consensus method, the malfunction of robot agent 2 directly affects the state values of the neighboring robot agents. Similarly in this method, the state value of each robot agent is sensitive to changes in its neighbor robot agents’ states; hence a large difference in state values between robot agents.

On the other hand, the proposed method shows that the wrong behavior of robot agent 2 and it is adapted successfully according to the desired user input. This is because each robot agent indirectly uses the configuration state of neighboring robot agents in adjusting its configuration state values by equation (1), which leads to be less susceptible to error conditions and more robust performance.

**Energy balancing adaptation.** To show the performance of energy efficiency and balance, we set the initial power ($E_0$) for each differently. The same error condition is applied randomly for all robot agents in the RSNS. Figure 3(a) shows the energy difference between robot agents with the proposed scheme and with the consensus scheme. The consensus method has no appropriate mechanisms for operation adaptation to schedule robot agent’s state, so the robot agents are activated stochastically. Each robot agent randomly generates a number and compares it with a given threshold. If the probability value is larger than the threshold, the robot agent becomes activated. In this simulation, the threshold values are set to 0.3 (mid-th), 0.02 (min-th), and 0.7 (max-th), respectively. For the metric of energy difference ratio, the smaller the energy difference is, the more the energy balance is achieved. The proposed method shows that the energy difference ratio decreases over time and goes near zero after 350 min. Therefore, energy consumption ratio is balanced among the robot agents in the RSNS. However, in the consensus method, the energy difference ratio is maintained or escalates. In the plot of the consensus method with min-th, the energy difference ratio suddenly drops to zero after 250 min, thus the robot agents’ batteries are exhausted. The energy depletion with min-th occurs within a shorter period as compared to other threshold values. This is caused by more frequent activation of robot agent. Figure 3(b) shows the residual energy ratio of the proposed and the consensus schemes. Result suggests the proposed method incurs the highest available
energy ratio; whereas, in the consensus method, energy depletion occurs relatively early, hence a reduction in network lifetime. The consensus method with min-th also shows a much shorter network lifetime than with other threshold values. In short, the proposed method automatically adjusts the operation state of each robot agent so energy consumption ratio is balanced among the robot agents. The proposed method can likewise avoid unnecessary energy waste and increase network lifetime effectively by automatically adjusting the operation state according to the configuration state.

Effects on varying control parameters. Figure 4(a) shows the effect of varying the value of $r_{\text{min}}$ on the residual energy ratio. Such an increase of $r_{\text{min}}$ reduces the residual energy ratio, hence a shorter network lifetime. In other words, if the value of $r_{\text{min}}$ is increased, although the steady state is entered, the active period is extended, resulting in unnecessary energy waste. Specifically, when the value of $r_{\text{min}}$ increases to 0.3, the residual energy ratio of the consensus method becomes somewhat higher than the proposed method. Therefore, the appropriate value of $r_{\text{min}}$ needs to be determined and adjusted for future works. Figure 4(b) shows the convergence time with different values of $\epsilon_r$. Convergence time is defined as the time when the difference between the value of operation state ($r_i$) and $r_{\text{min}}$ falls below 0.01. The convergence time decreases as the value of $\epsilon_r$ increases. However, for a smaller value of $\epsilon_r$, the operation state exactly converges to $r_{\text{min}}$; whereas, for higher value of $\epsilon_r$, the operation value continues to oscillate around $r_{\text{min}}$. Hence, choosing the optimal value of $\epsilon_r$ should also be studied in the future.
Conclusions

This paper presents a dual control approach for energy-efficient RSNS interaction system. First, for the system configuration control, the proposed scheme indirectly controls the consensus operation of the RSNS by propagating the configuration state values to the RSNS based on the proposed control laws of each robot agent. Second, we propose a controller for the robot agent’s operational state scheduling according to the configuration propagation rate. The proposed algorithm forwards the following major contributions. First, each robot agent automatically drives the system parameters to the desired system configuration even in an erroneous environment. Second, each robot agent effectively controls its operation mode according to the configuration state, thus balancing energy consumption in the RSNS. Finally, insights into the theoretical analysis of the proposed scheme are provided by deriving the system convergence and proving the system stability.

An important area for further study includes the selection of the values of parameter $r_{\text{min}}$ and $\epsilon_r$, estimation of performance impact of applied parameters, and state modeling of moving RSNS. This research could also investigate the effective collaboration among robot agents for appropriate decision making.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, Information and Communication Technology (ICT) and Future Planning (NRF-2020R1A2C1004390).

ORCID iD

Heejung Byun https://orcid.org/0000-0001-7693-8511

References

1. Barca J and Sekercioglu Y. Swarm robotics reviewed. *Robotica* 2013; 31(3): 345–359.
2. Brambilla M, Ferrante E, Birattari M, et al. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intell* 2013; 7(1): 1–41.
3. Tan Y and Zheng Z. Research advance in swarm robotics. *Defence Technol* 2013; 9(1): 18–39.
4. Ito S, Ohara K, Hoshi Y, et al. A robust formation control strategy for multi-agent systems with uncertainties via adaptive gain robust controllers. *Int J Eng Technol Innovation* 2021; 11(2): 71–87.
5. Zhang S, Staudinger E, Pöhlmann R, et al. Cooperative communication, localization, sensing and control for autonomous robotic networks. In: *Proceedings of the 2021 IEEE International conference on autonomous systems (ICAS)*, Montreal, QC, Canada, 11–13 August 2021, pp.1–5. New York: IEEE.
6. Yu X, Saldaña D, Shishika D, et al. Resilient consensus in robot swarms with periodic motion and intermittent communication. *IEEE Trans Rob* 2022; 38(1): 110–125.
7. Miao J, Li H, Zheng Z, et al. Secrecy energy efficiency maximization for UAV swarm assisted multi-hop relay system: joint trajectory design and power control. *IEEE Access* 2021; 9: 37784–37799.
8. Liekna A and Grundspenikis J. Towards practical application of swarm robotics: overview of swarm tasks. In: *Proceedings of the 2014 engineering for rural development*, 2014, pp.271–277. Jelgava: Latvia University of Agriculture.
9. Kolling A, Walker P, Chakraborty N, et al. Human interaction with robot swarms: a survey. *IEEE Trans Hum Mach Syst* 2016; 46(1): 9–26.
10. Kira Z and Potter M. Exerting human control over decentralized robot swarms. In: *Proceedings of the 2009 International conference on autonomous robots and agents*.
13. Croix J and Egerstedt M. A control lyapunov function approach to human-swarm interactions. In: Proceedings of the 2012 International conference on resilient control systems, Salt Lake City, UT, 14–16 August 2012, pp.197–202. New York: IEEE.

14. Goodrich M, Pendleton B and Kerman S. What types of interactions do bio-inspired robot swarms and flocks afford a human? In: Proceedings of the 2012 robotics, science, and systems, 2012, pp.105–112. doi: 10.15607/RSS.2012.VIII.014.

15. Gancet J, Motard E, Naghsh A, et al. User interfaces for human robot interactions with a swarm of robots. In: Proceedings of the 2014 IEEE international conference on robotics and automation, Anchorage, AK, 3–7 May 2010, pp.2846–2851. New York: IEEE.

16. Bashyal S and Venayagamoorthy G. Human swarm interaction for radiation source search and localization. In: Proceedings of the 2008 IEEE swarm intelligence symposium, St. Louis, MO, 21–23 September 2008, pp.1–8. New York: IEEE.

17. Miyakoshi K, Shun I, Hidetoshi O, et al. Synthesis of formation control systems for multi-agent systems under control gain perturbations. Adv Technol Innovation 2020; 5(2): 112–125.

18. Tahbien B, Lewis M, Lebiere C, et al. Towards a cognitively-based analytic model of human control of swarms. In: Proceedings of the 2014 AAAI spring symposium series, 2014, pp.68–73. Menlo Park: AAAI Press.

19. Nagavalli S, Luo L, Chakraborty N, et al. Neglect benevolence in human control of robotic swarm. In: Proceedings of the 2014 IEEE International conference on robotics and automation, Hong Kong, China, 31 May–6 June 2014, pp.6047–6053. New York: IEEE.

20. Sycara K, Lebiere C, Pei Y, et al. Abstraction of analytical models from cognitive models of human control of robotic swarms. In: Proceedings of the 2015 International conference on cognitive model, 2015. pp.13–18. ICCM 2015.

21. Walker P, Nunally S, Lewis M, et al. Neglect benevolence in human control of swarms in the presence of latency. In: Proceedings of the 2012 IEEE International conference on systems, man, and cybernetics, Seoul, Korea (South), 14–17 October 2012, pp.3009–3014. New York: IEEE.

22. Walker P, Amraii S, Lewis M, et al. Human control of leader-based swarms. In: Proceedings of the 2013 IEEE International conference on systems, man, and cybernetics, Manchester, 13–16 October 2013, pp.2712–2717. New York: IEEE.

23. Walker P, Amraii S, Chakraborty N, et al. Human control of robot swarms with dynamic leaders. In: Proceedings of the 2014 IEEE/RSJ International conference on intelligent robots and systems, Chicago, IL, 14–18 September 2014, pp.1108–1113. New York: IEEE.

24. Goodrich M, Pendleton B, Sujit P, et al. Toward human interaction with bio-inspired robot teams. In: Proceedings of the 2011 IEEE International conference on systems, man, and cybernetics, Anchorage, AK, 9–12 October 2011, pp.2859–2864. New York: IEEE.

25. Pendleton B and Goodrich M. Scalable human interaction with robotic swarms. In: Proceedings of the AIAA Infotech @ aerospace conference, 2015, pp.1–13. American Institute of Aeronautics and Astronautics. doi: 10.2514/6.2013-4731.

26. McLurkin J, Smith J, Frankel J, et al. Speaking swarmish: human-robot interface design for large swarms of autonomous mobile robots. In: Proceedings of the 2006 AAAI spring symposium, Technical Report SS-06-07, Stanford, California, USA, 27-29 March 2006, pp. 72–75.

27. Li M, Lu K, Zhu H, et al. Robot swarm communication networks: architectures, protocols, and applications. In: Proceedings of the 2008 International conference on communications and networking in China, Hangzhou, China, 25–27 August 2008, pp.162–166. New York: IEEE.

28. Chen M, Yang L, Kwon T, et al. Itinerary planning for energy-efficient agent communications in wireless sensor networks. IEEE Trans Veh Technol 2011; 60(7): 3290–3299.

29. Dmarogonas D, Frazzoli E and Johansson K. Distributed event-triggered control for multi-agent systems. IEEE Trans Automat Contr 2012; 57(5): 1291–1297.

30. Byun H. A method of indirect configuration with estimation of system state in networked multi-agent dynamic systems. IEEE Commun Lett 2018; 22(9): 1766–1769.

31. Meshabhi M and Egerstedt M. Graph theoretic methods in multi-agent networks. Princeton, NJ: Princeton University Press, 2010.