Comparing time. A New Approach To The Problem Of Time Synchronization In a Multi-agent System

Ivan Akinfiev
Saint Petersburg State University
Saint Petersburg, Russia
st121627@student.spbu.ru

Vlada Smetanina
Saint Petersburg State University
Saint Petersburg, Russia
vds.smetanina@gmail.com

Oleg Granichin
Laboratory “Control of Complex Systems”
IPME RAS
Saint Petersburg, Russia
o.granichin@spbu.ru

Ivan Arkhipov
Saint Petersburg State University
Saint Petersburg, Russia
arkhipov.iv99@mail.ru

***Abstract*—A live time-depended tasks execution based on distributed multi-agent network rely strongly on time synchronization ability of the system. A desynchronisation of the system may lead to incorrect result or even failure at performing a certain task. In this paper we propose a time-synchronization algorithm derived from fast Nesterov gradient descent.**

I. Introduction

Precise time synchronization is a non-trivial yet critical prerequisite for the correct and efficient operation of distributed multi-agent systems (MAS) [1], [2] , especially those deployed in applications with stringent timing requirements and dynamic operational contexts. Examples of such systems abound in modern engineering, encompassing domains such as cooperative robotics, sensor networks for environmental monitoring or surveillance, and distributed control systems for industrial automation or smart grids [3]. In these scenarios, where agents collaborate to achieve complex, time-sensitive objectives, even minor deviations in their individual time references can propagate through the network, leading to significant performance degradation, inaccuracies in data fusion and decision-making, task execution failures, and, in extreme cases, potential safety hazards. This inherent vulnerability underscores the paramount importance of developing robust and efficient time synchronization algorithms specifically tailored to mitigate the challenges posed by clock drift, network latency, and dynamic environmental conditions in distributed settings.

Traditional approaches to time synchronization [4], [5], [6] while offering valuable insights and foundational principles, often exhibit limitations when confronted with the complexities of real-world deployments. Consensus-based algorithms, for instance, may suffer from slow convergence rates, especially in large-scale networks or when faced with communication constraints. Clock synchronization protocols, such as the Network Time Protocol (NTP), can be susceptible to network delays and vulnerabilities, potentially impacting their accuracy and reliability. While Stochastic Gradient Descent (SGD) [7] has been explored as a potential solution for time synchronization, its performance can be suboptimal in dynamic environments [8]where clock drifts and measurement noise exhibit non-stationary characteristics. Consequently, there exists a pressing need for more sophisticated methods that can not only adapt to evolving conditions but also achieve precise synchronization rapidly, ensuring the reliable and timely execution of tasks within stringent time constraints.

Fast gradient methods, with Nesterov’s accelerated gradient descent [7] as a prime example, offer a compelling avenue for addressing these challenges. Unlike traditional gradient descent, which relies solely on the current gradient information, Nesterov’s method incorporates a ”momentum” term that leverages past gradient information to accelerate convergence, particularly for strongly convex functions. This inherent ability to exploit historical trends makes fast gradient methods well-suited for handling the dynamic nature of time synchronization, where clock drifts [9] and network conditions can vary over time. however, the direct application of fast gradient methods to time synchronization necessitates careful consideration of the unique challenges presented by distributed environments:

1) Non-stationarity: Individual clock drifts and fluctuating network conditions introduce time-varying parameters into the optimization problem, requiring algorithms that can adapt to this non-stationary optimization landscape. Traditional fast gradient methods, designed for static optimization problems, may struggle to maintain convergence in such dynamic settings.
2) Noise Resilience: Clock readings and communication channels are inherently noisy, introducing uncertainties that can hinder the convergence of optimization algorithms. Robustness to noise is therefore crucial to ensure accurate and reliable time synchronization.
3) Distributed Operation: Time synchronization algorithms must operate efficiently in a decentralized manner, taking into account limited communication bandwidth,

varying computational capabilities of agents, and the potential for network disruptions. This requires careful design choices to minimize communication overhead and ensure robust performance in the face of network imperfections.

This paper presents a novel time synchronization algorithm explicitly designed to address these challenges. By building upon the foundations of fast Nesterov gradient descent and incorporating mechanisms to handle non-stationarity and noise, our algorithm achieves both rapid and accurate time synchronization in distributed multi-agent systems. We provide a rigorous analysis of the algorithm's convergence properties, proving its ability to achieve synchronization under specific conditions. Furthermore, we demonstrate its effectiveness through simulations in realistic scenarios, showcasing its superior performance compared to existing methods. This work contributes to the advancement of reliable and high-performance time synchronization techniques, enabling the deployment of distributed MAS in increasingly complex and time-critical applications across diverse domains

II. Problem statement and proposed algorithm

A. Multi-agent network description

A network system consisting of j nodes called as agents is being considered. The system is working in a decentralized way and tries to synchronise a time flow of each agent to minimize a time difference between agents. Let $\mathcal{N} = \{1, ..., j\}$ be a set of agents that tries to synchronize a grid. Each agent has got their own working frequency ν^i, where i is a number of agent in $\mathcal{N}$. That makes an agent count it's own time as a first-order Markov process, from observer standpoint of view we can describe it as (1):

$$t^i_{k+1} = t^i_k + \frac{\omega^i_k}{\nu^i} + \zeta^i_k, \tag{1}$$

were ω_k is a time modification coefficient at step k, which will be modified by later proposed algorithm. For a single agent that is not connected to a network $z^i_k = 1$ at each k, ζ_k is a noise caused at step k by internal delays and both internal and external processes.

B. Agents communication

Agents are able to communicate with each other through a network described by the undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is a set of vertices and $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ is a set of edges. Let $(j, i) \in \mathcal{E}$ if there is an edge between agents j and i. The latter means that agent j is send his current time estimation to agent i and vice versa. For an agent $i \in \mathcal{N}$, the set of neighbors is defined as $\mathcal{N}^i = \{j \in \mathcal{N} : (j, i) \in \mathcal{E}\}$. The *in-degree* of $i \in \mathcal{N}$ equals $|\mathcal{N}^i|$. Here and after, $|\cdot|$ is the cardinality of a set.

We associate a weight $b^{ij} > 0$ with each edge $(j, i) \in \mathcal{E}$. Matrix $B = [b^{ij}]$ is called an adjacency or connectivity matrix of the graph. Denote $\mathcal{G}_B$ as the corresponding graph. Define the weighted *in-degree* of node i as the i-th row sum of B: $d^i(B) = \sum_{j=1}^n b^{ij}$ and $D(B) = \mathrm{diag}\{d^1(B), d^2(B), \dots, d^n(B)\}$ is the corresponding diagonal matrix. The symbol $\mathcal{L}(B) = D(B) - B$ stands for the Laplacian of graph $\mathcal{G}_B$.

In this paper, we rely on the next definition:

Definition 1. (**Connectivity**) An undirected graph $\mathcal{G}$ is said to be connected if there is a path between every pair of distinct vertices of $\mathcal{G}$.

This definition is widely used in cooperative networked systems [10] and related to consensus theory that will be used in our work.

C. Algorithm formulation

In this paper, the problem is to minimise the difference in time estimation between agents in a multi-agent system is being described. A synchronisation occurs at the time of consensus which for a simplicity happens every time-step of agent with a least frequency. There are 2 purposes of synchronisation: the first one is to understand how to modify the agents flow of time compared to other agents. It's done through the search of a coefficient $\omega_k = [\omega^1_k, ...\omega^j_k]^T$ using a consensus algorithm (2) that minimise a Laplacian potential [11].

Find $w_k = 1 - Argmin_{X_k \in \_R} X_k(t), \forall_k = 1, ..., k$

$$\omega_{k+1}(t) = 1 - \frac{1}{t^i_k h} \sum_{j \in N} b^{ij}(t^i_{k-1} + \frac{\omega_k}{\nu^i} - t^j_k), \tag{2}$$

where h is a step-size of consensus algorithm, t^j_k is time of j agent. Observation starts at $k = 0$ that makes algorithm applicable from $k = 1$ onwards.

Second one, is agents are trying to estimate a real time of a multi-agent system. We can also describe a system flow of time as a function of first-order Markov process (1) with noise.That allow to incorporate easily a loss function into combination with (2) to get a final function (3):

$$\begin{aligned} f(\hat{t}, t, \omega) = g \sum_{j \in N} (\hat{t_k} - t^j_k) \Big(p_1 - \frac{p_1 g}{t^i_k} \sum_{j \in N} (\hat{t_k} - t^j_k) + \\ + p_3 - \frac{p_3}{t^i_k h} \sum_{j \in N} b^{ij}(t^i_{k-1} + \frac{\omega_k}{\nu^i} - t^j_k) \Big), \end{aligned} \tag{3}$$

were $\hat{t}$ is a real time of a system, p_1, p_2, g are the weight coefficient of the function.

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be the underlying probability space corresponding to sample space Ω, set of all events $\mathcal{F}$, and probability measure $\mathbb{P}$. $\mathcal{F}_{k-1}$ be σ-algebra of all probabilistic events happened up to time instant $k = 1, 2, \dots$. $\mathbb{E}$ denotes mathematical expectation, and $\mathbb{E}_k$ denotes conditional mathematical expectation with respect to σ-algebra defined by $\hat{\mathbf{t}}_0, \dots, \hat{\mathbf{t}}_{k-1}$.

The algorithm proposed in this paper at each step $k = 1, ..., k_n$ (time starts at $k = 0$) provides estimate $\hat{\mathbf{t}}_k$ that can be use by the agent and trying to approach minimum $\mathbf{t}_k$ of the function (3) (minimum $\mathbf{t}_k$ can change over time). The algorithm provides a sequence of estimates $\{\hat{\mathbf{t}}_k\}_{k=0}^{\infty}$ solving the following problem (4):

Find $\hat{\mathbf{t}}_k$ s.t. $\exists N, C < \infty$: $\forall k > K$

$$\mathbb{E}_k \|\hat{\mathbf{t}}_k - \mathbf{t}_k\|^2 \le C. \tag{4}$$

Assumption 1. Functions F_k have a common Lipschitz constant $L > 0$ and strong convexity constant $\mu > 0$:

$$\forall \mathbf{t} \in \mathbb{R}^n \quad \|\nabla F_k(\mathbf{t})\| \le L\|\mathbf{t} - \mathbf{t}_k\|, \tag{5}$$

$$\langle \nabla F_k(\mathbf{t}), \mathbf{t} - \mathbf{t}_k \rangle \ge \mu \|\mathbf{t} - \mathbf{t}_k\|^2. \tag{6}$$

Assumption 2. For every $k \ge 0$, drift is bounded as

$$\|F_k(\mathbf{t}) - F_{k+1}(\mathbf{t})\| \le a\|\nabla F_k(\mathbf{t})\| + b, \tag{7}$$

$$\|\nabla F_{k+1}(\mathbf{t}) - \nabla F_k(\mathbf{t})\| \le c. \tag{8}$$

Assumption 3. $\forall i \in \mathcal{N}, j \in \mathcal{N}^i$ the noises $\xi_k^{i,j}$ are centered and have bounded variance σ^2.

Assumption 4. Communication graph $\mathcal{G}_B$ is connected.

To solve the problem defined by (3) with consideration of Assumptions 1-4 we propose the Fast Local Voting Protocol for Time Synchronisation algorithm (FLVPTS) for each agent i. Algorithm strongly relies on the our result we previously obtained in [12]. The proposed FLVPTS algorithm has following steps:

1) Chose $\hat{t}_0^i \in \mathbb{R}$. (if time starts at 0 seconds chose 0). Set $t_o^i = \hat{t}_0^i$. Chose $h > 0$, $\eta \in (0, \mu)$, $\alpha_x \in (0, 1)$ so that α_x satisfying the inequality (9) can always be found. Define $H_1 = h - \frac{h^2 L}{2}$.
2) k-th iteration ($k \ge 0$):
   a) Find $\alpha_k \in [\alpha_x, 1)$ so that
   $$H_1 - \frac{\alpha_k^2}{2\gamma_{k+1}} > 0. \tag{9}$$
   b) Let $\gamma_{k+1} = (1 - \alpha_k)\gamma_k + \alpha_k(\mu - \eta)$.
   c) Choose
   $$z_k^i = \frac{1}{\gamma_k + \alpha_k(\mu - \eta)}(\alpha_k \gamma_k t_k^i + \gamma_{k+1}\hat{t}_k^i)$$
   and find $f^i(z_k^i, t, \omega)$.
   d) Find a new estimate $\hat{t}_{k+1}^i = z_k^i + h f^i(z_k^i, t, \omega)$.
   e) Set t_{k+1}^i as (10):
   $$t_{k+1}^i = \frac{1}{\gamma_k}\Big[(1 - \alpha_k)\gamma_k t_{k+1}^i + \\ +\alpha_k(\mu - \eta) z_k^i - \alpha_k f^i(z_k^i)\Big]. \tag{10}$$

The proposed algorithm should be launched at each agent i independently.

Theorem 1. The problem (3) is solved by the FLVPTS algorithm with

$$C = \frac{2}{\mu} D_\infty$$

where D_∞ is defined as (11):

$$D_\infty = \alpha_\star^{-1}\Big[\frac{2a + hc}{4\epsilon} + 2b + \\ + (1 - \alpha_\star)(b + A_\infty c) + \\ + h^2 \frac{L}{2}\sigma^2 + \frac{c^2}{2\eta}\Big] \tag{11}$$

for $\Gamma = \max_{n \ge 0} \gamma_k$, $\epsilon \in \Big(0, \frac{1}{a(1+\alpha_\star) + hc}\Big(H - \frac{\alpha_\star^2}{2\Gamma}\Big)\Big]$ and $\alpha_\star$, η, h chosen in the algorithm.

The estimation error after a finite number of iterations is bounded as:

$$\mathbb{E}_k f_k(\hat{t}_k) - f_k^\star \le \prod_{i=1}^{n}(1 - \alpha_k)(\phi_0(t_0) - f^\star + \Phi) + D_k$$

where $\phi_0(x) = f_0(\hat{t}_0) + \frac{\gamma_0}{2}\|x - t_0\|^2$, $\Phi = \frac{\gamma_0 c^2}{2\mu^2}$, $\{\alpha_k\}_{n=0}^\infty$, $\{\Lambda_k\}_{n=0}^\infty$ and $\{Z_k\}_{n=0}^\infty$ are sequences defined as (12-14):

$$\alpha_k \in [\alpha_\star, 1), \quad \Lambda_0 = 1, \quad \Lambda_{k+1} = (1 - \alpha_k)\Lambda_k \tag{12}$$

$$A_0 = 0, \quad A_{k+1} = (1 - \alpha_k)((1 - \lambda_k)a + A_k), \\ Z_k = (1 - \lambda)(b + ac) + A_k c, \tag{13}$$

$$D_0 = 0, \quad D_{k+1} = (1 - \alpha_k)D_k + \frac{a(1 + \alpha_k) + hc}{4\epsilon} + \\ + (1 + \alpha_k)b + (1 - \alpha_k)Z_k + \\ + h^2 \frac{L}{2}\sigma^2 + \frac{\alpha_k c^2}{2\eta}. \tag{14}$$

Proof is mainly similar to that in [12].

III. Result of simulations

In this section, we present a result of a numerical experiment which illustrates the synchronisation between 5 fully connected agents with non-equal rate of work. The synchronization process takes place after a certain amount of time (15 seconds) has passed since the start of the experiment. Synchronization occurs at a frequency that is equal to the frequency of an agent with a lowest rate of work. Parameters of algorithm define as follows:$L = 7,1, \mu = 0,9, \nu = 0,8, \gamma = 0,08, \alpha = 0,08$. These parameters are chosen to satisfy Assumptions 1–4.

The quality of protocol is defined by 2 metrics:

1) RMSE of agent i compared to other agents at time-step k (Fig.1):
$$RMSE_k^i = \sqrt{\frac{1}{N-1}\sum_{j=1}^{N}(\hat{t}_k^i - \hat{t}_k^j)^2}$$
2) Mean $RMSE_k$ at time step k (Fig. 2):
$$MRMSE_k = \frac{1}{N}\sum_{i=1}^{N} STD_k^i$$

At Fig.3 we can observe the time flow of each agent with and without system synchronization compared to the simulation of global time flow (in a way described as a ground truth). The difference between the 8 Hz agent is due to the too low frequency of operation, this agent was introduced to see how the algorithm works with agents operating at low frequencies.

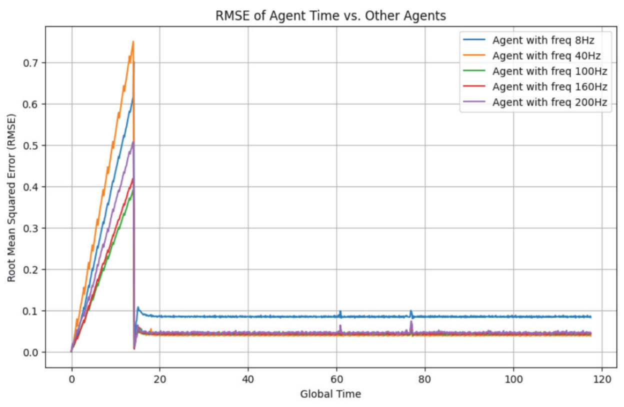

Fig. 1. RMSE of agent i

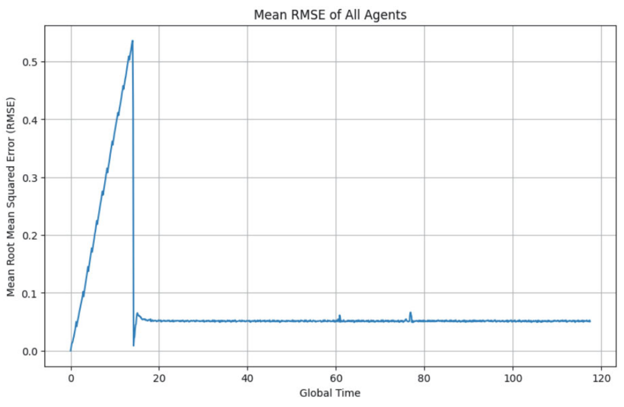

Fig. 2. Mean $RMSE_k$ at time step k

IV. Conclusion

This article presented a time synchronization algorithm for multi-agent systems, suitable for synchronizing time-dependent tasks such as live process simulations. Several avenues for further development are possible. Adaptive parameter tuning could be implemented at each step, employing reinforcement learning and neural networks to optimize performance. Additionally, incorporating the ability for the algorithm to recognize the need for synchronization within the context of the current task could enhance efficiency. Finally, exploring methods for correcting past time steps during synchronization could improve overall accuracy and consistency.

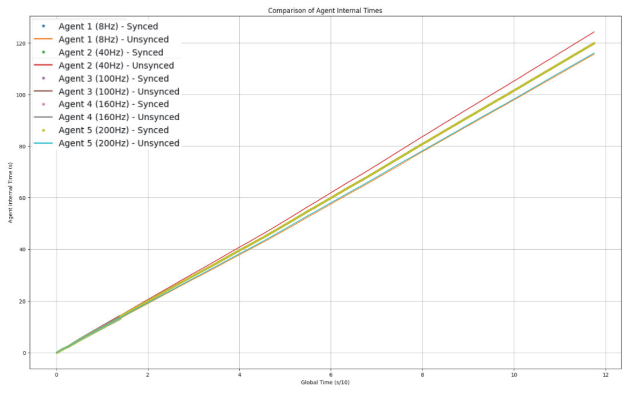

Fig. 3. The time flow of each agent

References

[1] H. F. Ahmad, "Multi-agent systems: overview of a new paradigm for distributed systems," in *7th IEEE International Symposium on High Assurance Systems Engineering, 2002. Proceedings.* IEEE, 2002, pp. 101–107.

[2] X. Ge, Q.-L. Han, D. Ding, X.-M. Zhang, and B. Ning, "A survey on recent advances in distributed sampled-data cooperative control of multi-agent systems," *Neurocomputing*, vol. 275, pp. 1684–1701, 2018.

[3] K. M. Muttaqi, A. E. Nezhad, J. Aghaei, and V. Ganapathy, "Control issues of distribution system automation in smart grids," *Renewable and Sustainable Energy Reviews, vol. 37, pp. 386–396, 2014.*

[4] A. Amirkhani and A. H. Barshooi, "Consensus in multi-agent systems: a review," *Artificial Intelligence Review*, vol. 55, no. 5, pp. 3897–3935, 2022.

[5] R. Mart´ınez-Guerra, C. D. Cruz-Ancona, and C. A. P´erez-Pinacho, "Generalized multi-synchronization viewed as a multi-agent leader-following consensus problem," *Applied Mathematics and Computation*, vol. 282, pp. 226–236, 2016.

[6] Y. Li and C. Tan, "A survey of the consensus for multi-agent systems," *Systems Science & Control Engineering*, vol. 7, no. 1, pp. 468–482, 2019.

[7] Y. Nesterov, "A method for solving the convex programming problem with convergence rate o (1/k2)," in *Dokl akad nauk Sssr*, vol. 269, 1983, p. 543.

[8] H. A. Abbas, S. I. Shaheen, and M. H. Amin, "Organization of multi-agent systems: an overview," *arXiv preprint arXiv:1506.09032*, 2015.

[9] B. Simons, "An overview of clock synchronization," *Fault-Tolerant Distributed Computing*, pp. 84–96, 2005.

[10] F. L. Lewis, H. Zhang, K. Hengster-Movric, and A. Das, *Cooperative control of multi-agent systems: optimal and adaptive design approaches*. Springer Science & Business Media, 2013.

[11] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on automatic control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[12] D. Kosaty, A. Vakhitov, O. Granichin, and M. Yuchi, "Stochastic fast gradient for tracking," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 1476–1481.