2023 International Wireless Communications and Mobile Computing (IWCMC) | 979-8-3503-3339-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/IWCMC58020.2023.10182856

Assessing Distributed Consensus Performance on Mobile Cyber-Physical System Swarms

Jiayi Chen, Maxime Vacheron, Bruno Chianca Ferreira and Guthemberg Silvestre
ENAC, Université de Toulouse, France

jiayi.chen@recherche.enac.fr, maxime.vacheron@alumni.enac.fr, {bruno.chianca, silvestre}@enac.fr

***Abstract*—Mobile Cyber-Physical System (CPS) Swarms are likely to change our daily lives significantly in several application domains, including smart cities, space exploration, environmental monitoring, and transporting systems. Yet, industrial projects fundamentally rely on centralized communication infrastructures which in turn lead to suboptimal performance and limited autonomy for both sensing and actuating. To enhance communication and distributed coordination within the swarm, reliable distributed computing through consensus protocols constitutes a promising approach. Nonetheless, only a handful of studies has considered evaluating reliable distributed computing in Mobile CPS Swarms. The goal of this work is to evaluate the performance of consensus protocols on CPS swarms with processing units deployed on mobile nodes. Running on top of an emulation framework for mobile CPS, our preliminary evaluation focuses on evaluating three key deployment aspects of Mobile CPS Swarms executing a consensus protocol: the effective cost of mobility; the impact of the connectivity of nodes and the eventual network partitions; and the price of continuous routing updates in the mobile environment. Our experimental results indicate the transient network partition in mobile environments are common. They also suggest that the sparsity of dynamic communication network and continuous routing updates have a major impact on the performance of distributed consensus protocols.**

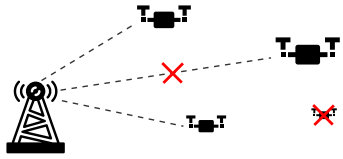
***Index Terms*—Fault-tolerant mobile computing, Distributed consensus, Paxos, Cyber-physical systems, Swarm intelligence.**

I. Introduction

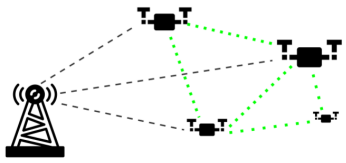
We are currently experiencing the revolution of industry *4.0* in terms of Cyber-Physical Systems (CPS). According to the NIST definition [17], "Cyber-physical systems are engineered systems that are built from and depend upon the synergy of computational and physical components". Unlike traditional embedded systems, which are designed as stand-alone devices, CPS focuses more on networking several devices [31]. At the same time, by using integrated sensors to realize real-time haptics and support customized actuation on physical devices, they provide high-quality user personalized services and experiences by reducing communication delays [5]. There has been an upward trend in having information and services everywhere at hand, thus these emerging systems are inevitable in the highly networked world of today.

CPS have already been used in many projects of academia and industry such as smart electric power grids, manufacturing systems, aerospace systems, and defense systems [36]. By leveraging engineered swarm system and mobility, groups of CPS (e.g., swarm of drones, underwater robotics, nanosat constellations or ground rovers) have an opportunity to interact with each other in order to collectively provide swarm intelligence [39] through adaptability, robustness, and scalability. However, the design and the operations of CPS swarms remain exceptionally challenging, which constitute major obstacles to a successful transition from research platforms to industrial swarm applications [40].

Indeed, many industrial projects still rely on centralized infrastructure to pragmatically circumvent challenges in control, data sharing and processing atop of a CPS swarm. Whereas centralized approaches might speed-up the design and the deployment of industrial prototypes, they hinder the ability of CPS swarms to achieve high autonomy and fault tolerance. Figure 1(a) shows a centralized approach commonly used to operate a swarm of drones [21]. It illustrates a CPS swarm where mission's control, data collection, data sharing and processing strongly rely on the central base station, which, in turn, results in poor fault tolerance and unpredictable performance issues. Alternatively, Figure 1(b) depicts a swarm of CPSs that relies on a distributed, dynamic configuration service to tolerate faults and to improve performance through swarm-level coordination and synchronisation.



(a) Centralized infrastructure



(b) Distributed infrastructure

Fig. 1. Examples of CPS swarm based on a different infrastructure.

In this work, we argue that fault-tolerant, distributed computing is key to enhance both autonomy and fault tolerance in CPS swarms. In this context, distributed consensus protocols contribute to enabling swarm intelligence by providing fault-tolerant sub-systems including dynamic reconfiguration,

distributed locking, reliable broadcast and state machine replication. Although consensus protocols have been extensively studied [9], [12], [20], [22], [30], [32], [37], [43], [44], previous works mostly overlooked the performance evaluation and practical aspects of consensus algorithms in mobile systems in general and in mobile CPS swarms in particular [35]. For instance, it is well known that wireless communication eventually increases message loss, and that network partitions of the communication network are far more likely to occur in mobile environments [45]. Consequently, wireless communication and mobility undoubtedly create serious uncertainties that affect the functioning of consensus protocols [38]. Yet, the impact of concurrent, practical uncertainties on the performance of consensus protocols in mobile CPS swarms remains unclear.

Therefore, our goal is to better understand the impact of intrinsic uncertainties of CPS swarms on consensus protocols. Specifically, we aim at evaluating the performance of a real-world implementation of multi-Paxos [42], by far the most popular consensus protocol in production systems. Using an emulation framework for mobile CPS [15], we investigate the impact of three practical, common uncertainties created by the mobile environment of CPS swarms: (1) the effective cost of mobility of the nodes within the deployed mission's area, (2) the impact of the connectivity of nodes and the eventual network partitions, and (3) cost of continuous routing updates in the mobile environment.

Overall, our work makes the following contributions:

- We survey fault-tolerant, distributed computing in CPS swarms on mobile environments. First we explain the important role of distributed computing on fault-tolerant, autonomous CPS swarms. Then we introduce mobile networks for CPS and we describe how consensus protocols can be useful for emerging CPS swarms, including nanosatellite constellations and swarm of unmanned aerial vehicle (UAV, also known as drones);
- We conduct experiments to evaluate the performance of a real implementation of a consensus protocol in an emulated, mobile CPS swarm. Our preliminary results quantify the critical trade-off between fault tolerance and performance. They also suggest that both the sparsity of the communication graph and continuous routing updates considerably hinder the performance of consensus protocols in mobile CPS swarms.

The remainder of this work is organized as follows: We first provide some background about distributed computing on CPS swarms and consensus protocols in mobile environments in Section II. Then we present the system model in Section III. Section IV describes our experimental settings. Section V contains a set of preliminary experiments assessing the performance of the consensus protocol multi-Paxos in mobile CPS swarms. We discuss the limitations of our work and the relevant topics in Section VI. Finally, we conclude our work in Section VII.

II. Background

We first provide some background on distributed computing on Cyber-Physical Systems (CPS) swarms. Then, we briefly review consensus protocols in mobile environments.

A. Distributed computing on CPS swarms

CPS are industrial automation systems that enable many innovative functionalities through their networking and their access to the cyber world [24]. Such systems find applications in a number of large-scale, safety-critical domains, such as satellite constellation, swarm of drones, smart cities, and smart grids [4]. In a CPS swarm, multiple nodes cooperate to provide services whose performance and availability strongly rely on swarm's properties, such as fault tolerance and autonomy [39].

The increasingly large amounts of data that must be collected, processed and transmitted by emerging CPS applications challenges the design and the deployment of CPS swarms. Despite the powerful computation capacity of cloud computing platforms, the limited transmission capacity of remote CPS nodes and the high propagation delay to push data to cloud infrastructures inevitably usually incur prohibitive performance degradation [7]. Yet, the most of the state-of-the-art CPS swarms still relies on centralized operations techniques, which in turn constitute a major obstacle to a successful transition from research platforms to industrial swarm applications [40]. We argue that distributed computing [41] can be leveraged to improve both fault tolerance and autonomy of emerging CPS swarms. Indeed, distributed computing on CPS can decrease the dependency on third-party, cloud service providers by enabling fault-tolerant, distributed services.

A good example of an emerging CPS systems that can fully take advantage of distributed computing is a nanosatellite constellation. In a constellation, nanosats must continuously synchronise sensing units and computing tasks, potentially for multiple application workloads, colocated on the same constellation. To improve both mission efficiency and fault tolerance, nanosats could directly exchange messages via wireless communication links to accomplish the system targets in a synchronized, distributed manner [33]. Consequently, although nanosats constellation would still need to eventually transmitting data to ground stations under the bent-pipe architecture, a distributed orbital edge computing solution [13] would improve constellation's autonomous reconfiguration and synchronisation. For instance, it would contribute to coping with partial failures of constellation's nanosats, enhancing missions' deployment and operations, as well as to reducing the latency and to increasing throughput of the constellations' distributed services. Similarly, swarms of drones [10] can leverage distributed computing to offer fault-tolerant, novel services. For instance, a swarm of drones can rely on an distributed, dynamic configuration service to continuously maintain a consistent list of available sensors and actuators, or they can use distributed locking service to perform a conflict-free, cooperative sensing [19].

B. Consensus protocols in mobile environments

Distributed consensus is a fundamental building block of many practical, fault-tolerant computing applications, including distributed storage systems [34], wait-free coordination services and dynamic configuration for real-world production systems [6], [23]. Broadly speaking, the distributed consensus involves getting a set of processes to agree on a value proposed by one or more of the processes [11]. To guarantee the distributed consensus, a consensus protocol implements a fault-tolerant algorithm that formally enforces specific safety and liveness properties with regard to predefined timing and failure assumptions [14]. Among the production distributed consensus services, multi-Paxos [42] protocol, based on Paxos [28] algorithm, is by far the most popular. In a nutshell, Paxos algorithm works with two types of nodes to constitute a strongly consistent and fault-tolerant replica set, namely leader and replica. Clients issue requests to the leader. In order to respond a client's request, the leader spreads it to the replicas, and waits for a majority of replicas' replies. While a majority of replicas is available, the replica set consistently processes clients' requests. Although distributed consensus was widely evaluated in wired, cloud-based systems, only a handful of previous work investigated distributed consensus in wireless networks [35] and in CPS swarms [5].

There exist two major types of mobile wireless networks: infrastructure-based networks and mobile ad-hoc networks (MANETs). MANET technology is perhaps unique in its ability to facilitate collaboration among mobile users that have no fixed infrastructures for communication support [3]. Although some probabilistic protocols for MANETs have been proposed, few works have been reported on performance analysis of consensus protocols. We are interested in understanding the performance of consensus protocols in mobile networks with properties similar to MANETs. In order to reproduce additional features specific to CPS systems, this work relies on MACE [15], an emulation framework for mobile CPS.

Unlike traditional wired networks, mobile environments in general have dynamic properties regarding the communication network, node's mobility, resources' availability and capacity, which make the design and deployment of distributed algorithms much more difficult [45]. In addition, intrinsic characteristics of mobile networks introduce additional challenges and uncertainties to be coped by consensus protocols, including unpredictable messages loss and network partitions. It is important to note that consensus protocols trade either availability or network partitions off for enforcing strong consistency of the replicated data, as described in CAP theorem [18]. Therefore, both of the aforementioned uncertainties, common in mobile environments, are likely to render the consensus protocol unavailable. To the best of our knowledge, this is the first study of the impact of practical, intrinsic uncertainties of CPS swarms, such as mobility and network density, on the performance of the consensus protocols.

III. System Model

We consider a distributed system composed by two non-overlapping sets of processes, a finite set of N nodes $S = \{s_1, s_2, \ldots, s_N\}$ and an infinite set of clients $C = \{c_1, c_2, \ldots\}$. Each process has a unique identifier. Every client or server knows the set of nodes. The processes communicate by message passing. The consensus protocol investigated in this work is considered in a CPS swarm that consists of a set of M CPS nodes, including N nodes that constitute a replica set for solving the consensus problem. Moreover, the consensus protocol runs under the following assumptions.

Failure model. Processes may fail by crashing, but do not experience arbitrary behaviour (i.e., no Byzantine failures). The network is mostly unreliable and is subject to unpredictable latencies, as well as load imbalances (e.g., peak demand), that imposed on both nodes and the network. Such imbalances may cause variations in transmission delays. However, nodes rely on reliable one-to-one communication channels, where transmitted messages can be lost (and retransmitted) but not corrupted. Therefore, the system tolerates f faulty nodes such that $f < \lceil N/2 \rceil$ (a quorum exists).

Timing model. We assume that the system is partially synchronous [14], that is, it is initially asynchronous and eventually becomes synchronous, because a consensus protocol cannot be both safe and live under asynchronous assumptions, as the FLP impossibility result [16] states.

IV. Experimental Settings

We assume the consensus protocol runs atop of N replicas (a replica set), such that $M \geq N$, where M is the total number of mobile CPS nodes. Each node has a communication range fixed to 160m, which means in-range pairs of nodes can exchange messages directly. Out-of-range pairs of nodes can eventually communicate to each other if there exists (at least) a reachable route between them. Routes of a mobile CPS swarm are continuously updated with B.A.T.M.A.N. IV [1], a real-world routing protocol for highly dynamic MANETS [25]. Figure 2 is captured from the GUI (graphical user interface) of the emulator, which clearly shows how nodes communicate with each other. We take the node at the figure's center as an example, the two in-range nodes can exchange messages with it directly (solid blue lines), the other two out-of-range nodes communicate with it by the reachable route.

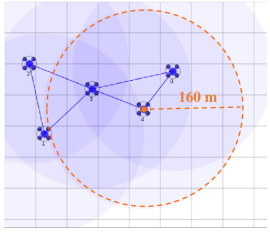


Fig. 2. Example of communication topology.

For generality, we make no assumption about the mobility pattern of CPS swarm's nodes. Therefore, to make experiments

reproducible, nodes move at slightly varying speeds ranging from 2 to 2.1 meters/second according to the *random waypoint* (RWP) mobility model [26]. For simplicity, RWP model emulates random movements for each mobile node in a two-dimensional (2D) square area of size A. We define the network density D as

$$D = \frac{M}{A} \tag{1}$$

for a number of nodes M and an area size A.

To reproduce such a mobility within A, our experiments rely on the Mobile Ad-hoc Computing Emulator [15], MACE, an emulation framework for mobile CPS swarms. MACE generates a virtual environment as close to reality as possible that allows us to run and evaluate distributed consensus implementations. It is worth noting that the nodes' mobility is likely to incur transient network partitions, mostly caused by routing protocol updates and out-of-range events in the mobile environment.

We use Paxi [2], a framework that provides real-world implementation of (multi-)Paxos. For ease of explanation and to focus on the practical uncertainties of mobile environments, we assume that the leader is fixed and it is co-located with a set of emulated clients and a replica in a single node. For instance, when $N = M = 3$, there are three nodes, including a leader L [1] and two replicas($\{R_1, R_2\}$).

To assess the performance of consensus protocol, we consider two well-known system-level metrics: throughput (in requests per second) and latency (in milliseconds). To measure the scalability in terms of tail latency, we instrument Paxi to sequentially issue an increasing load of concurrent read/write requests from the aforementioned set of clients up to the system saturation. Finally, each experiment was repeated five times to report performance statistics (i.e., median and 95th percentile). Table I summarizes the main parameters of our experimental settings.

TABLE I
MAIN PARAMETERS IN EXPERIMENTAL SETTINGS.

No. of CPS swarm nodes	M
No. of replica set nodes (Paxos-specific)	N ($M \geq N$)
No. of tolerated faulty nodes (Paxos-specific)	f ($N = 2f + 1$)
Read/write rate (Paxos-specific)	0.5
Total number of keys (Paxos-specific)	1000
Area size	A
Network density	$D = M/A$
Communication range	160 meters
Routing protocol	B.A.T.M.A.N IV [25]
Bandwidth	433.3Mbps
Delay	3000us
Packet loss	0%
Running time of an experiment	100 seconds
Mobility model	Random waypoint
Moving speed	2-2.1m/s

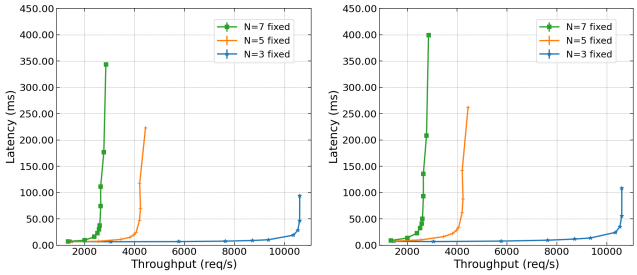
[1]Note that the leader L also plays the role of replica in the replica set.

V. PERFORMANCE EVALUATION

The objective of the performance evaluation is to study the performances of consensus protocol – multi-Paxos on a swarm of CPSs, which is a specific mobile ad-hoc network. We perform the first set of experiments with fixed nodes in order to establish a baseline. After that we conduct three sets of experiments to study the impact of mobility, the effect of nodes connectivity, and the cost of continuous routing updates.

A. Baseline experiment with fixed nodes

The first set of experiments involves fixed nodes and is subjected to determine a baseline evaluation. In the baseline measurements, we varied the number of replica as follows: $N = M = \{3, 5, 7\}$, and we assumed a perfect deployment scenario, where fixed nodes are close enough to each other to always ensure optimal, one-hop communications. The results are plotted in Figure 3.



(a) Median latency (b) 95th percentile latency

Fig. 3. Baseline evaluation with fixed nodes.

We observe that all the curves reach a maximum throughput suggesting the saturation of replica set. We also observe a big performance gap as the size of replica set increases. For instance, a CPS swarm of three replicas can answer more than 10000 requests per second, while a configuration with seven replicas can answer around 3000 requests per second only.

When the number of replicas (N) increases, the resilience of the system against faulty nodes f also increases. However, this results in larger quorums, consequently the replica set needs to spend more time reaching an agreement, which incurs in higher latencies and lower throughput.

B. Cost of mobility

The second set of experiments aims at assessing the impact of mobility on the performance of a consensus protocol. For this experiment, we varied the number of replicas, $N = M = \{3, 5, 7\}$, and we considered nodes are able to move within a two-dimensional square zone with a network density of 80 nodes/km^2.

We compare the results in mobile CPS swarms with the baseline of Section V-A by plotting both in Figure 4. For readability, plots in the rest of the paper, depict the median latency only [2].

[2]With mobile nodes, the performance variability of 95 percentile latency measurements became too high, therefore they were omitted.

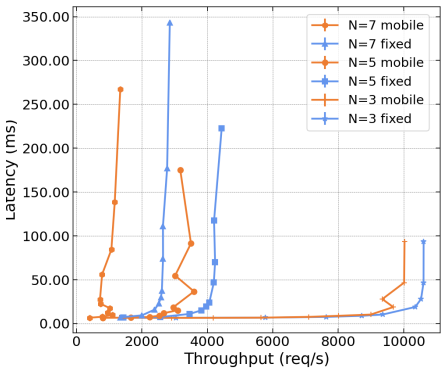


Fig. 4. The cost of mobility in a consensus protocol.

Regardless of the configuration, i.e., $N = \{3, 5, 7\}$, our results suggest that nodes' mobility leads to severe performance degradation. In fact, as transient network partitions occurs, replica sets running consensus protocols, like Paxos, become temporally unavailable, which in turn results in performance losses. Our results indicate that transient network partitions in mobile environments are common.

In addition, we observe that the higher the fault tolerance f, the bigger the performance gap between the baseline and the mobile CPS swarm. For instance, when $f = 1$ ($N = 3$), the performance losses due to mobility is nearly 6%. The performance gap is about 60% when $f = 3$ ($N = 7$). These findings clearly highlight the well-known trade-off between performance and fault tolerance of consensus protocols, providing worthwhile insight for mobile CPS swarms configurations. For simplicity, the results presented in the rest of this work are obtained using $f = 2$ ($N = 5$) as a moderate level of fault tolerance for better mobile swarm autonomy.

C. Impact of connectivity of nodes

To gain a deeper understanding of how consensus protocols perform in mobile environments, we conducted experiments to examine the impact of connectivity of nodes. Therefore, for the same number of replicas $N = M = 5$, we varied the 2D square area to reproduce three notable classes network density (obtained via the Equation 1) detailed in Table II [3]. Our preliminary findings suggest that a relative high number of nodes (i.e., at least 41 nodes/km^2) is required to run a consensus protocol properly.

TABLE II
NETWORK DENSITY SETTINGS.

Area size A	Network density D	Density class
150×150 m^2	222 nodes/km^2	High density
250×250 m^2	80 nodes/km^2	Medium density
350×350 m^2	41 nodes/km^2	Low density

[3]It is worth noting that two lower densities were evaluated, 25 nodes/km^2 and 16 nodes/km^2, but they were omitted because they performed very poorly due to too frequent transient network partitions.

Figure 5 summarizes the impact of network density on consensus protocols in a mobile CPS swarm. It is important to note that the results are worse and more unstable when the network density decreases. For instance, the protocol's throughput drops by nearly 90% as the network density changes from high to low. In addition, we found that 38% test files fail to run when the low density is considered. This is primarily due to frequent network partitions caused by low density, which results in the unavailability of the replicated state machine (RSM) that relies on five replicas.

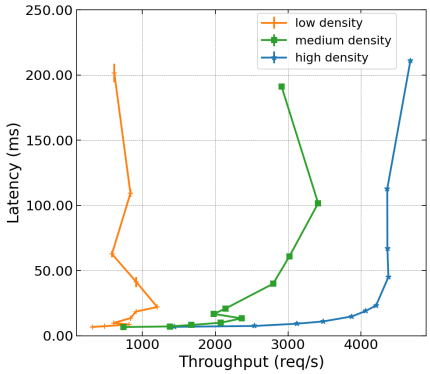
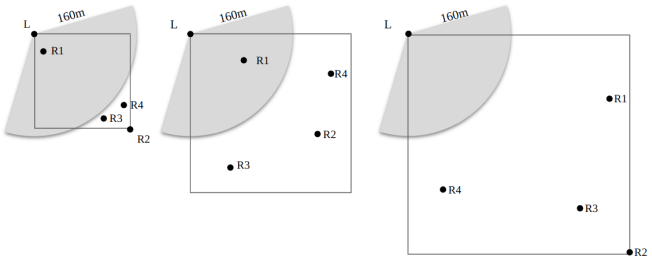


Fig. 5. Impact of connectivity of nodes.

To better explain the above results, Figure 6 shows three examples of worst-case scenarios within different area size that result in out-of-range events when $N = 5$ (including leader L and four other replicas, $\{R_1, R_2, R_3, R_4\}$). In all these cases, a quorum of three nodes, including the leader L and any 2 other replicas, is required to reach consensus and to make progress. Suppose the leader L is located in one corner of square area and there is only one in-range (i.e., located inside the L's gray shaded area) node, as depicted in Figures 6(a) and (b), thus there is no quorum and the replica set becomes temporally unavailable to ensure consistency of the previously executed requests, as the CAP theorem [18] states. Similarly, Figure 6(c) shows a worst-case scenario where the leader is completely isolated.



(a) 150m×150m (b) 250m×250m (c) 350m×350m

Fig. 6. Three worst-case scenarios examples.

In general, the network density decreases with increasing node movement area, in which case chances that no quorum with the leader is possible at any moment of the execution due to mobility and wireless communications uncertainties. As a result, the nodes' connectivity reduces and the overall performance of the system degrades, incurring higher latencies and lower throughput. Indeed, the frequency of these practical, transient incidents are closely related to network density and they are likely to result in transient network partitions and, in turn, severe performance degradation.

D. Effect of continuous routing updates

To investigate the cost of continuous routing updates in mobile environment, we considered mobile CPS swarms whose missions cover increasingly bigger areas. Therefore, whenever a CPS node is added to the swarm, it routes messages of the replica set nodes to i) permit the swarm to cover a bigger area and to keep the same network density; and to ii) eventually improve the fault tolerance of the replica set [4]. To this end, we fixed $N = 5$ and the network density to a medium level (i.e., 80 nodes/km^2), then we incrementally increased the number of nodes M from 5 to 10. By increasing M for the same N, the number of routing updates also increases. As a result, the number of hops to exchange messages of the consensus protocol among replica set nodes is likely to increase. Figure 7 summarizes the experimental results.

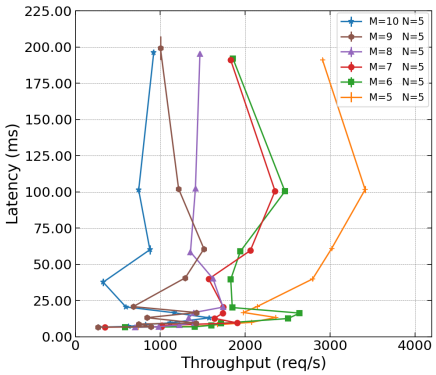


Fig. 7. Impact of continuous routing updates.

Our results suggest that routing updates cause a consistent, large performance degradation of the consensus protocol. Although one would expect higher latencies as the consensus protocol's messages need to cross more nodes, the cost of continuous routing updates seem prohibitively high. Additionally, we observed that as M increases, the proportion of failed executions using Paxi also increases, as depicted in Table III. For instance, for $M = 10$, the communication within the replica set is so poor that half of experiment executions simply failed.

[4]For instance, while $M > N$, we assume that additional nodes can eventually replace faulty replicas in a system reconfiguration.

TABLE III
PERCENTAGE OF FAILED EXECUTIONS USING PAXI FOR DIFFERENT VALUES OF M.

The value of M	5	6	7	8	9	10
Percentage of failed executions	3.6%	7.3%	16.4%	20%	23.6%	50.9%

VI. DISCUSSION

While our preliminary work paves the way to better understand the performance of consensus protocols on mobile CPS swarms, our current study has several known limitations. We believe that addressing these limitations will permit obtaining a deeper understanding of the performance of distributed consensus on mobile CPS swarms.

Consensus protocol framework and mobile wireless systems. Paxi provides a convenient open-source framework to benchmark real-world implementation of (multi-)Paxos, and it is particularly useful for independent reproducibility. However, the number of failed executions using Paxi were surprisingly high as the number of mobile replicas was increased. These findings emphasize the need for further investigation to understand and adapt Paxi to mobile wireless systems, including considering additional Paxos protocol variants.

Fault tolerance and blockchains. For simplicity, our current failure model only assumes crash faults, where a mobile node simply stops all computation and communication. Yet, CPS nodes exchanging messages through a wireless network may be subject to a wide variety of non-crash faults, where a node acts arbitrarily. Thus, our failure model could be improved in order to cope with Byzantine faults [29]. This research direction would also contribute to the study of blockchains and smart contracts [8], [27] on mobile CPS swarms, since blockchains rely on consensus protocols to ensure all participants agree on a unified transaction ledger [46]. Nonetheless, it is important to note that the use of blockchains in CPS swarms is ultimately complementary to our work.

VII. CONCLUSION

The emerging applications for mobile CPS swarms will allow us to provide complex, future industrial services. Yet, the design and the operations of CPS swarms remain exceptionally challenging, which constitute major obstacles to a successful transition from research platforms to industrial swarm application. We argue that a distributed consensus protocol is key to enable essential features on these promising CPS swarm applications, especially autonomy and fault tolerance.

In this work, we present a preliminary performance evaluation of the most common consensus protocol in production systems, Paxos. Different from previous work, our study focuses on practical aspects of a real implementation of a consensus protocol in mobile CPS swarms, including mobility, nodes' connectivity and continuous routing updates. Our experimental results atop of an emulation framework for mobile CPS suggest that mobility, network density and continuous routing updates have a major impact on the performance of a consensus protocol. In addition, we assert that the mobile

deployment trade-offs are not obvious and require further investigation: whenever the size of the replica set is bigger than five or much smaller than the total number of CPS nodes, severe performance degradations were observed.

REFERENCES

[1] B.a.t.m.a.n, 2021. https://open-mesh.org/projects/batman-adv/wiki.

[2] Ailidani Ailijiang, Aleksey Charapko, and Murat Demirbas. Dissecting the performance of strongly-consistent replication protocols. In *Proceedings of the 2019 International Conference on Management of Data*, pages 1696–1710, 2019.

[3] Khaled Alekeish and Paul Ezhilchelvan. Consensus in sparse, mobile ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, 23(3):467–474, 2011.

[4] Alessandra Bagnato, Regina Krisztina Bíró, Dario Bonino, Claudio Pastrone, Wilfried Elmenreich, René Reiners, Melanie Schranz, and Edin Arnautovic. Designing swarms of cyber-physical systems: the h2020 cpswarm project. In *Proceedings of the Computing Frontiers Conference*, pages 305–312, 2017.

[5] Umesh Bodkhe, Dhyey Mehta, Sudeep Tanwar, Pronaya Bhattacharya, Pradeep Kumar Singh, and Wei-Chiang Hong. A survey on decentralized consensus mechanisms for cyber physical systems. *IEEE Access*, 8:54371–54401, 2020.

[6] Mike Burrows. The chubby lock service for loosely-coupled distributed systems. In *Proceedings of the 7th symposium on Operating systems design and implementation*, pages 335–350, 2006.

[7] Kun Cao, Shiyan Hu, Yang Shi, Armando Walter Colombo, Stamatis Karnouskos, and Xin Li. A survey on edge and edge-cloud computing assisted cyber-physical systems. *IEEE Transactions on Industrial Informatics*, 17(11):7806–7819, 2021.

[8] Ethan Cecchetti, Fan Zhang, Yan Ji, Ahmed Kosba, Ari Juels, and Elaine Shi. Solidus: Confidential distributed ledger transactions via pvorm. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 701–717, 2017.

[9] Saksham Chand and Yanhong A Liu. What's live? understanding distributed consensus. *arXiv preprint arXiv:2001.04787*, 2020.

[10] Wu Chen, Jiajia Liu, and Hongzhi Guo. Achieving robust and efficient consensus for large-scale drone swarm. *IEEE Transactions on Vehicular Technology*, 69(12):15867–15879, 2020.

[11] George F Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed systems: concepts and design*. pearson education, 2005.

[12] Xavier Défago, André Schiper, and Péter Urbán. Total order broadcast and multicast algorithms: Taxonomy and survey. *ACM Computing Surveys (CSUR)*, 36(4):372–421, 2004.

[13] Bradley Denby and Brandon Lucia. Orbital edge computing: Nanosatellite constellations as a new class of computer system. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 939–954, 2020.

[14] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM (JACM)*, 35(2):288–323, 1988.

[15] Bruno Chianca Ferreira, Guillaume Dufour, and Guthemberg Silvestre. Mace: A mobile ad-hoc computing emulation framework. In *2021 International Conference on Computer Communications and Networks (ICCCN)*, pages 1–6. IEEE, 2021.

[16] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985.

[17] National Science Foundation(NSF). Cyber-physical systems, 2012. https://www.nsf.gov/pubs/2021/nsf21551/nsf21551.htm.

[18] Seth Gilbert and Nancy Lynch. Perspectives on the cap theorem. *Computer*, 45(2):30–36, 2012.

[19] Alessandro Giusti, Jawad Nagi, Luca Gambardella, and Gianni A Di Caro. Cooperative sensing and recognition by a swarm of mobile robots. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 551–558. IEEE, 2012.

[20] Vassos Hadzilacos and Sam Toueg. A modular approach to fault-tolerant broadcasts and related problems. Technical report, Cornell University, 1994.

[21] Gautier Hattenberger, Murat Bronz, and Michel Gorraz. Using the paparazzi uav system for scientific research. In *IMAV 2014, international micro air vehicle conference and competition 2014*, pages pp–247, 2014.

[22] Heidi Howard and Richard Mortier. Paxos vs raft: Have we reached consensus on distributed consensus? In *Proceedings of the 7th Workshop on Principles and Practice of Consistency for Distributed Data*, pages 1–9, 2020.

[23] Patrick Hunt, Mahadev Konar, Flavio P Junqueira, and Benjamin Reed. {ZooKeeper}: Wait-free coordination for internet-scale systems. In *2010 USENIX Annual Technical Conference (USENIX ATC 10)*, 2010.

[24] Nasser Jazdi. Cyber physical systems in the context of industry 4.0. In *2014 IEEE international conference on automation, quality and testing, robotics*, pages 1–4. IEEE, 2014.

[25] David Johnson, Ntsibane S Ntlatlapa, and Corinna Aichele. Simple pragmatic approach to mesh routing using batman. 2008.

[26] David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile computing*, pages 153–181. Springer, 1996.

[27] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE symposium on security and privacy (SP)*, pages 839–858. IEEE, 2016.

[28] Leslie Lamport. Paxos made simple. *ACM SIGACT News (Distributed Computing Column) 32, 4 (Whole Number 121, December 2001)*, pages 51–58, 2001.

[29] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. In *Concurrency: the works of leslie lamport*, pages 203–226. 2019.

[30] Butler W Lampson. How to build a highly available system using consensus. In *International Workshop on Distributed Algorithms*, pages 1–17. Springer, 1996.

[31] Edward A Lee. Cyber physical systems: Design challenges. In *2008 11th IEEE international symposium on object and component-oriented real-time distributed computing (ISORC)*, pages 363–369. IEEE, 2008.

[32] L Leslie. The part-time parliament. *ACM Transactions on Computer Systems*, 16(2):133–169, 1998.

[33] Guo-Ping Liu and Shijie Zhang. A survey on formation control of small satellites. *Proceedings of the IEEE*, 106(3):440–457, 2018.

[34] John Ousterhout, Parag Agrawal, David Erickson, Christos Kozyrakis, Jacob Leverich, David Mazières, Subhasish Mitra, Aravind Narayanan, Diego Ongaro, Guru Parulkar, et al. The case for ramcloud. *Communications of the ACM*, 54(7):121–130, 2011.

[35] Valentin Poirot, Beshr Al Nahas, and Olaf Landsiedel. Paxos made wireless: Consensus in the air. In *EWSN*, pages 1–12, 2019.

[36] Ragunathan Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical systems: the next computing revolution. In *Design automation conference*, pages 731–736. IEEE, 2010.

[37] Sergio Rajsbaum and Michel Raynal. 60 years of mastering concurrent computing through sequential thinking. *ACM SIGACT News*, 51(2):59–88, 2020.

[38] Michel Raynal. *Fault-tolerant message-passing distributed systems: an algorithmic approach*. Springer, 2018.

[39] Melanie Schranz, Gianni A Di Caro, Thomas Schmickl, Wilfried Elmenreich, Farshad Arvin, Ahmet Şekercioğlu, and Micha Sende. Swarm intelligence and cyber-physical systems: concepts, challenges and future trends. *Swarm and Evolutionary Computation*, 60:100762, 2021.

[40] Melanie Schranz, Martina Umlauft, Micha Sende, and Wilfried Elmenreich. Swarm robotic behaviors and current applications. *Frontiers in Robotics and AI*, 7:36, 2020.

[41] Andrew S Tanenbaum. Distributed operating systems. *CERN European Organization for Nuclear Research-Reports-CERN*, pages 101–108, 1996.

[42] Robbert Van Renesse and Deniz Altinbuken. Paxos made moderately complex. *ACM Computing Surveys (CSUR)*, 47(3):1–36, 2015.

[43] Robbert Van Renesse, Nicolas Schiper, and Fred B Schneider. Vive la différence: Paxos vs. viewstamped replication vs. zab. *IEEE Transactions on Dependable and Secure Computing*, 12(4):472–484, 2014.

[44] Michael Whittaker, Neil Giridharan, Adriana Szekeres, Joseph Hellerstein, and Ion Stoica. Sok: A generalized multi-leader state machine replication tutorial. *Journal of Systems Research*, 1(1), 2021.

[45] Weigang Wu, Jiannong Cao, Jin Yang, and Michel Raynal. Design and performance evaluation of efficient consensus protocols for mobile ad hoc networks. *IEEE Transactions on Computers*, 56(8):1055–1070, 2007.

[46] Yang Xiao, Ning Zhang, Wenjing Lou, and Y. Thomas Hou. A survey of distributed consensus protocols for blockchain networks. 22(2):1432–1465.