# Network Synthesis for Tactical Environments: Scenario, Challenges, and Opportunities

Tzanis Anevlavis[a], Jonathan Bunton[a], Jared Coleman[b], Mine Gokce Dogan[a], Eugenio Grippo[b], Abel Souza[c], Christina Fragouli[a], Bhaskar Krishnamachari[b], Matthew Maness[d], Karl Olson[d], Prashant Shenoy[c], Paulo Tabuada[a], and Gunjan Verma[d]

[a]University of California, Los Angeles, USA
[b]University of Southern California, USA
[c] University of Massachusetts Amherst, USA
[d]US Army's CCDC Army Research Laboratory, USA

## ABSTRACT

We develop a network synthesis scenario, which is built around a concrete perimeter surveillance application, yet we believe captures a number of the challenges and requirements that are common to other tactical communication and computational network applications. The proposed scenario addresses the problem of binary population identification within a perimeter: our goal is to synthesize a sensing and computing network that classifies people moving within a given perimeter into one of two categories (e.g., friend or foe). We discuss several open challenges that we organize across the following clusters: sensor placement, communication network provisioning and optimization, computational task placement, dynamic re-synthesis and resilience under adversarial settings. We also briefly discuss approaches that attempt to address such challenges.

## 1. INTRODUCTION

The proliferation of battlefield sensors can enable the execution of various warfighting functions (such as target detection, identification, localization, etc.) at unprecedented speed and scale. Recent advances. in particular on the Internet-of-Things (IoT), have made it possible to envision the deployment of networks containing not only sensors, actuators, and compute points, but even autonomous mobile robots and UAV's to support tactical applications such as perimeter surveillance, search and rescue, and more.

A key challenge in such systems, that is often unexplored or taken for given, is the initial synthesis of such tactical IoT networks. Yet network synthesis is very important, since sensing, communication and inference all strongly depend and build on the underlying network topology. Moreover, unlike civilian applications where network synthesis can be planned well in advance and rely on the availability of high quality hardware, fast computational units, or fat communication pipes, in military operations we may have a fast developing situation and need to acquire situational awareness over an area very quickly - using very limited resources. For instance, we may not have electricity and need to rely on batteries; we may have limited memory; we may need to operate within a hostile environment where communication channels, computational nodes and sensors may get destroyed. Further, in a dynamic operating environment, there is an associated challenge of re-synthesizing the network, as objectives change or as new resources are added/removed/damaged or operating conditions such as network bandwidth and availability change.

In this paper, we argue that making progress on the various challenges associated with network synthesis and re-synthesis requires the definition of some common operating scenarios that could be potentially used as a benchmark for research efforts. In this work, we contribute the definition of one such scenario pertaining to binary object recognition within a defined perimeter.

The paper is organized as follows. After defining the network synthesis scenario in Section 2, we describe some of the many challenges that arise in this setting and that need to be addressed in future research in Section 3. These include challenges associated with sensor placement, network provisioning and optimization, computational task placement, dynamic re-synthesis, and resilience under adversarial settings.

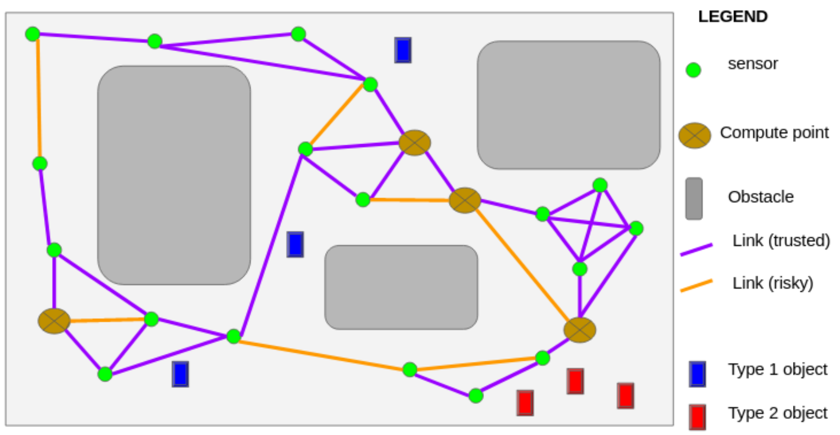

Figure 1. Illustration of network synthesis scenario pertaining to in-perimeter object detection and binary classification

## 2. SCENARIO DEFINITION

Our proposed scenario models the problem of binary object identification within a perimeter and is illustrated in Figure 1. We assume that a map of the area within this perimeter is provided to us, where some areas are obstructed and we need to cover the remaining "free" area with sensing and computation resources. Our goal is to synthesize a sensing and computing network that classifies objects (which could be humans or vehicles) moving within a given perimeter to one of two categories (e.g., friend or foe). In Figure 1, these are the Type 1 (blue) and Type 2 (red) objects.

We consider both static and mobile sensors that range from low-cost simple devices to more capable ones augmented with computational capabilities. We depict sensors with green circles in Figure 1 - we consider only static sensors for simplicity of illustration. The data from each sensor is sent to one or more classification algorithms for processing. Each such algorithm runs on a computation unit, depicted as a brown compute point in Figure 1. The sensors may connect to one or more compute points (algorithms); moreover, we may have side information whether some communication links can be trusted (purple), or are risky (orange) which we may want to take into account in our designs. Each classification algorithm is characterized by several parameters such as data requirements, memory requirements, processing requirements, and quality of the produced output. These algorithms, in turn, inform an application that builds and keeps an updated occupancy map indicating where individuals of each class are present within the perimeter.

In a real-world implementation of this scenario, it would be necessary to further specify hardware and software availability, as well as requirements on how they may need to be configured together to meet the over-all goal. There could also be additional specified environmental conditions that can impact network synthesis beyond obstacles, such as physical constraints on device placement.

Synthesizing a network for this scenario would be evaluated through metrics that benchmark solutions. These could include both network-level metrics such as throughput, latency, energy-usage, cost, and risk as well as overall application-level metrics such as the accuracy, sensitivity, and specificity of the binary object classification.

## 3. OPEN CHALLENGES

At a bird's eye view, the open questions include where to place heterogeneous sensors, computing, and network resources; how to configure and provision the network and route data; where to place the classification and occupancy algorithms; how to approach multi-objective optimization goals; how to distribute computational tasks; how to gracefully re-synthesize the network in the presence of mobile nodes or perimeter updates; and how to protect against and operate under adversarial attacks. We next discuss in more detail some of these questions and provide indicative references to related work.

## 3.1 Sensor Placement

One key aspect of the network synthesis problem is the ability to quickly synthesize the location of sensors to maximize sensing quality and coverage while being mindful of constraints imposed by the need to use the network to offload data for processing. Moreover, algorithms for network synthesis need to be fast since changes in tactical environments may require the network to be quickly re-synthesized, and need to be flexible to potentially incorporate already deployed assets.

Traditionally, sensor placement is addressed using fixed sensing region models for each sensor and then using geometry to cover a maximal number of locations of interest, or by applying randomized placements with pre-specified distributions, which are effective in particular sensing scenaria.[1,2] Similarly, typical communications models for wireless sensor networks are disc models or binary connectivity conditions, which may be poor models in practice.[3] Modern applications require new and diverse sensing devices demanding more flexible and accurate sensing and communication models. Thus, a main question becomes, how to formulate the network synthesis as an optimization problem, that allows for fast and efficient solution, and can potentially also provide optimality guarantees.

In formulating an optimization problem, we can consider one of two approaches: formulate a continuous optimization problem where the location of each sensor is described through continuous coordinate variables, or partition the area (for instance in grid squares) and formulate a discrete optimization problem that specifies which squares are occupied by sensors and which are not. Each of these approaches has advantages and disadvantages; for instance, the continuous formulation may make more challenging the formulation of some of desired constraints, such as bounding the number of hops to a compute unit; while the discrete optimization problem may make more challenging aspects such as computational efficiency. An approach to address the second is provided in.[4] This approach leverages submodularity and generalizes the usual approach to submodular function maximization with near-optimal sensing utility guarantees in.[5] Moreover, it allows for a richer class of constraints, when compared to the standard method in[6] that handles single connectivity and cardinality constraints. From a performance perspective, the proposed approach compares favorably to the state of the art iterated submodular knapsack algorithm in[7] as indicated by the experiments. Additionally, in[8] we can find an approach addressing the first and an attempt at merging the two using mixed-integer linear programming.

In both cases, there are a number of associated interesting questions that might be worth exploring, that include the following: (i) can we derive theoretical performance guarantees; (ii) although our optimization problem may output a specific location, the actual deployment would be approximate; what is the sensitivity to such displacements; (iii) how to incorporate robustness to adversarial attacks; (iv) how to achieve a deployment that may serve more than one mission objective; (iv) what is the scalability of the proposed algorithms; (v) how to efficiently re-synthesize when the perimeter area slightly changes (e.g. new obstacles appear, slight variation of the area covered occurs); and (v) how to effectively deploy potentially mobile sensors.

## 3.2 Communication Network Provisioning and Optimization

Protocols for wireless communication for sensor networks have been well studied in the literature; a prevalent approach is to maintain a (dynamic) tree that connects the sensors to the sink, see for instance.[9–14] Additional techniques, such as using network coding to increase robustness, have also been explored.[15]

However, the problem of how to synthesize a wireless network, in terms of topology placement, is less well understood. Consider the case where a source and a destination have a fixed position and we want to see where to place relays to support their communication. What is the optimal topology seems to depend on what are the communication capabilities of the sensors; for instance, if physical layer cooperation is possible, the optimal topologies can be very different than when it is not.[16] Similarly, whether sensor nodes can leverage broadcasting or not, or how much state they can keep, can affect what is the optimal topology.

In our particular setup, we do not have a single source communicating with a single destination, but instead we have multiple sources (sensors) communicating with a single destination (sink); the open questions in this case include: (i) if we have the possibility to add "helper" relay nodes, that would support the communication, where should these be placed? (ii) if we can also slightly alter the location of the sources, so that the sensing

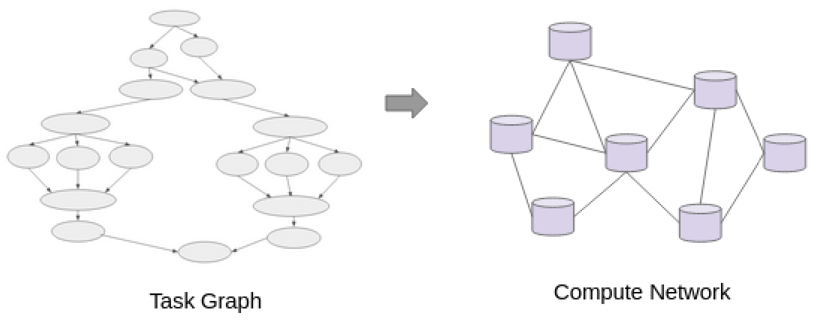

Figure 2. Dispersed computing enables the mapping and scheduling of complex applications expressed as task graphs on to a compute network

performance degrades within a given margin, can we design efficient algorithms that do so and achieve significant benefits in terms of connectivity?

As part of any algorithm that decides where to place nodes to optimize for communication, we need to be able to efficiently calculate min-cut values. Even though a min-cut value calculation between a single source and a single destination are polynomial in the size of the network,[17–22] for a sensor network with multiple sources, we need to be able to calculate min-cut values between any subset of the sources and the destination - thus an exponential number of calculations. How to achieve this efficiently is an open problem. Moreover, if broadcasting or physical layer cooperation is enabled, we need to be able to calculate information theoretic min-cut values efficiently; first steps towards this are explored in,[23] but this is also an area not well understood.

Finally, the question of re-synthesis, when for instance communication links get blocked or fail, and we have the possibility to not only alter how we route the information over the existing communication network, but also potentially to add new nodes, or change the position of existing ones, is also not well explored. Data-driven methods, such as the approach we tried in[24] that uses reinforcement-learning, could potentially help in this direction.

## 3.3 Computational Task Placement

When the computational application pertaining to processing the raw sensor data into actionable information is more complex, it may be represented in the form of a computational task-graph. Mapping the task graph on to a given set of networked computing resources is a challenging problem, traditionally arising in the context of grid computing, and more recently addressed in the domain of dispersed computing (see figure 2). Classical scheduling algorithms such as HEFT,[25] randomized algorithms,[26] evolutionary algorithms,[27] and newer variants[28–31] have been designed to optimize the placement of tasks in particular nodes, while accounting for data dependencies, with respect to metrics such as makespan and throughput. Recent work has focused on developing a framework called Jupiter, that enables orchestration of containerized tasks over heterogeneous hybrid edge-cloud resources and dynamic networks.[32]

Additional considerations include the placement of computations onto hardware accelerators such as neural accelerators or edge GPUs in order to improve the efficiency and speed of computations.[33–35] Algorithms for multiplexing task computations onto a set of edge accelerators, which may be possibly heterogeneous, are needed while respecting the resource and communication requirements of tasks.[36]

From a network synthesis perspective, an important question to explore in this connection is how to configure and provision a tactical network that enables efficient mapping of computation on to the available resources while balancing other considerations such as cost and risk. Recent work has started to address this question through a general multi-objective network synthesis framework for dispersed computing.[37]

### 3.4 Dynamic re-synthesis

The placement of computational tasks onto edge nodes will need to be re-synthesized periodically to handle dynamics such as workload changes, change in resource requirements, or change in user requirements.[36] While a full synthesis of the task graph can be performed using the above algorithms, an incremental approach where the placement of only a subset of the tasks is modified based on the dynamics is more efficient and faster. Such an approach will need to determine what subset of the tasks are no longer seeing good performance and determine how to efficiently move them to new edge nodes with adequate resources so as to meet application's end-to-end requirements. There has been some work in the task graph scheduling problem for networks that are dynamic due to faulty nodes,[38, 39] changes to the task graph or network (i.e. newly deployed compute-nodes, performance degredation, etc.),[40–42] compute-node mobility, and/or differences between planning and runtime environments.[43]

A related question is how to perform this dynamic re-synthesis in the presence of unknown stochastic or otherwise dynamic links. For unknown stochastic links, recent work has explored the use of multi-armed bandit formulations and corresponding algorithms for this problem.[44] While some existing cloud systems are equipped with schedulers that adapt to changes in communication strength between network nodes,[45–48] the problem of re-synthesizing a network to support a specific application has received less attention in the literature.

### 3.5 Resilience under Adversarial Settings

In a tactical environment, going beyond performance optimization, network synthesis must also explicitly consider providing resilience against an adversary.[49] The adversary may seek to disrupt network operations through active jamming, by injecting fake traffic (including false sensor readings), eavesdropping, or by damaging sensors, compute points and network relays.[50] Securing against such adversarial attacks requires first of all the application of traditional and innovative solutions to provide confidentiality (both for stored and in-transit data as well as computations, using techniques such as homomorphic encryption or secure hardware including trusted execution environments), integrity (through the use of hashing and other cryptographic primitives and systems) and availability (e.g. resilience to jamming and DDoS attacks). Further, it also requires careful introduction of redundancy during network synthesis to maximize the ability to survive and sustain network operations, such as by ensuring diverse multi-path connectivity between key nodes or by applying Byzantine fault tolerant consensus approaches that can tolerate arbitrary failures of subsets of nodes. Another class of techniques to improve resilience to adversarial attacks is to incorporate mobile nodes (which could be robots or UAVs), tasked to ensure that the network is configured/reconfigured as needed.[51, 52]

## ACKNOWLEDGMENTS

This work was supported in part by the Army Research Laboratory under Cooperative Agreement W911NF-17-2-0196.

## REFERENCES

[1] Gupta, H., Zhou, Z., Das, S., and Gu, Q., "Connected sensor cover: self-organization of sensor networks for efficient query execution," *IEEE/ACM Transactions on Networking* **14**(1), 55–67 (2006).

[2] Younis, M. and Akkaya, K., "Strategies and techniques for node placement in wireless sensor networks: A survey," *Ad Hoc Networks* **6**(4), 621–655 (2008).

[3] Al-Karaki, J. N. and Gawanmeh, A., "The optimal deployment, coverage, and connectivity problems in wireless sensor networks: Revisited," *IEEE Access* **5**, 18051–18065 (2017).

[4] Bunton, J., Anevlavis, T., Verma, G., Fragouli, C., and Tabuada, P., "Split to win: near-optimal sensor network synthesis via path-greedy subproblems," in [*MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM)*], 789–794 (2021).

[5] Zhang, H. and Vorobeychik, Y., "Submodular optimization with routing constraints.," in [*AAAI*], **16**, 819–826 (2016).

[6] Krause, A., Guestrin, C., Gupta, A., and Kleinberg, J., "Near-optimal sensor placements: Maximizing information while minimizing communication cost," in [*Proc. 5th Int. Conf. on Information Processing in Sensor Networks*], 2–10 (2006).

[7] Iyer, R. and Bilmes, J., "Submodular optimization with submodular cover and submodular knapsack constraints," in [*Advances in Neural Information Processing Systems*], 2436–2444 (2013).

[8] Ghosh, P., Bunton, J., Pylorof, D., Vieira, M. A. M., Chan, K. S., Govindan, R., Sukhatme, G., Tabuada, P., and Verma, G., "Synthesis of large-scale instant IoT networks," *IEEE Transactions on Mobile Computing* , 1–1 (2021).

[9] Gnawali, O., Fonseca, R., Jamieson, K., Moss, D., and Levis, P., "Collection tree protocol," in [*Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*], *SenSys '09*, 1–14, Association for Computing Machinery, New York, NY, USA (2009).

[10] Khan, I., Belqasmi, F., Glitho, R., Crespi, N., Morrow, M., and Polakos, P., "Wireless sensor network virtualization: A survey," *IEEE Communications Surveys Tutorials* **18**(1), 553–576 (2016).

[11] El Emary, I. M. M. and Ramakrishnan, S., [*Wireless Sensor Networks: From Theory to Applications*], CRC Press, Inc., USA (2013).

[12] Khan, M., Pandurangan, G., and Anil Kumar, V., "Distributed algorithms for constructing approximate minimum spanning trees in wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems* **20**(1), 124–139 (2009).

[13] Reinhardt, A., Morar, O., Santini, S., Zöller, S., and Steinmetz, R., "Cbfr: Bloom filter routing with gradual forgetting for tree-structured wireless sensor networks with mobile nodes," in [*2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*], 1–9 (2012).

[14] Luo, D., Zhu, X., Wu, X., and Chen, G., "Maximizing lifetime for the shortest path aggregation tree in wireless sensor networks," in [*2011 Proceedings IEEE INFOCOM*], 1566–1574 (2011).

[15] Keller, L., Atsan, E., Argyraki, K., and Fragouli, C., "Sensecode: Network coding for reliable sensor networks," *ACM Trans. Sen. Netw.* **9** (apr 2013).

[16] Dogan, M. G., Ezzeldin, Y. H., and Fragouli, C., "On optimal relay placement in directional networks," in [*2021 IEEE International Symposium on Information Theory (ISIT)*], 3156–3161 (2021).

[17] Ford, D. R. and Fulkerson, D. R., [*Flows in Networks*], Princeton University Press, USA (2010).

[18] Gomory, R. and Hu, T. C., "Multi-terminal network flows," *Journal of the Society for Industrial and Applied Mathematics* **9**, 551–570 (Dec. 1961).

[19] Hariharan, R., Kavitha, T., Panigrahi, D., and Bhalgat, A., "An $\tilde{\mathrm{O}}$(mn) gomory-hu tree construction algorithm for unweighted graphs," in [*Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*], *STOC '07*, 605–614, Association for Computing Machinery, New York, NY, USA (2007).

[20] Li, J. and Panigrahi, D., [*Approximate Gomory–Hu Tree is Faster than n – 1 Max-Flows*], 1738–1748, Association for Computing Machinery, New York, NY, USA (2021).

[21] Abboud, A., Krauthgamer, R., and Trabelsi, O., [*Subcubic Algorithms for Gomory–Hu Tree in Unweighted Graphs*], 1725–1737, Association for Computing Machinery, New York, NY, USA (2021).

[22] Hanifehnezhad, S. and Dolati, A., "Gomory hu tree and pendant pairs of a symmetric submodular system," in [*Topics in Theoretical Computer Science - Second IFIP WG 1.8 International Conference, TTCS 2017, Tehran, Iran, September 12-14, 2017, Proceedings*], Mousavi, M. R. and Sgall, J., eds., *Lecture Notes in Computer Science* **10608**, 26–33, Springer (2017).

[23] Dogan, M. G., Ezzeldin, Y., and Fragouli, C., "Gomory-hu trees over wireless," *IEEE Communications Letters* , 1–1 (2022).

[24] Dogan, M. G., Ezzeldin, Y. H., Fragouli, C., and Bohannon, A. W., "A reinforcement learning approach for scheduling in mmwave networks," in [*MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM)*], 771–776 (2021).

[25] Topcuoglu, H., Hariri, S., and Wu, M.-Y., "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE transactions on parallel and distributed systems* **13**(3), 260–274 (2002).

[26] Binato, S., Hery, W., Loewenstern, D., and Resende, M. G., "A grasp for job shop scheduling," in [*Essays and surveys in metaheuristics*], 59–79, Springer (2002).

[27] Zhu, Z., Zhang, G., Li, M., and Liu, X., "Evolutionary multi-objective workflow scheduling in cloud," *IEEE Transactions on parallel and distributed Systems* **27**(5), 1344–1357 (2015).

[28] Hu, D. and Krishnamachari, B., "Throughput optimized scheduler for dispersed computing systems," in [*2019 7th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*], 76–84, IEEE (2019).

[29] Zhang, W., Zhang, Z., Zeadally, S., and Chao, H.-C., "Efficient task scheduling with stochastic delay cost in mobile edge computing," *IEEE Communications Letters* **23**(1), 4–7 (2018).

[30] Bittencourt, L. F., Sakellariou, R., and Madeira, E. R., "Dag scheduling using a lookahead variant of the heterogeneous earliest finish time algorithm," in [*2010 18th Euromicro conference on parallel, distributed and network-based processing*], 27–34, IEEE (2010).

[31] Singh, K., Alam, M., and Sharma, S. K., "A survey of static scheduling algorithm for distributed computing system," *International Journal of Computer Applications* **129**(2), 25–30 (2015).

[32] Ghosh, P., Nguyen, Q., Sakulkar, P. K., Tran, J. A., Knezevic, A., Wang, J., Lin, Z., Krishnamachari, B., Annavaram, M., and Avestimehr, S., "Jupiter: a networked computing architecture," in [*Proceedings of the 14th IEEE/ACM International Conference on Utility and Cloud Computing Companion*], 1–8 (2021).

[33] Lima, J. V., Gautier, T., Danjean, V., Raffin, B., and Maillard, N., "Design and analysis of scheduling strategies for multi-cpu and multi-gpu architectures," *Parallel Computing* **44**, 37–52 (2015).

[34] Bleuse, R., Kedad-Sidhoum, S., Monna, F., Mounié, G., and Trystram, D., "Scheduling independent tasks on multi-cores with gpu accelerators," *Concurrency and Computation: Practice and Experience* **27**(6), 1625–1638 (2015).

[35] Raravi, G. and Nélis, V., "A ptas for assigning sporadic tasks on two-type heterogeneous multiprocessors," in [*2012 IEEE 33rd Real-Time Systems Symposium*], 117–126, IEEE (2012).

[36] Liang, Q., Hanafy, W. A., Ali-Eldin, A., and Shenoy, P., "Model-driven cluster resource management for ai workloads in edge clouds," *In review. Available as Arxiv preprint 2201.07312* (2022).

[37] Coleman, J., Grippo, E., Krishnamachari, B., and Verma, G., "Multi-objective network synthesis for dispersed computingin tactical environments," *Conference on Signal Processing, Sensor/Information Fusion, and Target Recognition XXXI, part of SPIE Defense + Commercial Sensing* (2022).

[38] Kurt, M. C., Krishnamoorthy, S., Agrawal, K., and Agrawal, G., "Fault-tolerant dynamic task graph scheduling," in [*SC'14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*], 719–730, IEEE (2014).

[39] Yu, Z. and Shi, W., "An adaptive rescheduling strategy for grid workflow applications," in [*2007 IEEE International Parallel and Distributed Processing Symposium*], 1–8, IEEE (2007).

[40] Cao, Z., Zhou, L., Hu, B., and Lin, C., "An adaptive scheduling algorithm for dynamic jobs for dealing with the flexible job shop scheduling problem," *Business & Information Systems Engineering* **61**(3), 299–309 (2019).

[41] Hao, X., Dai, Y., Zhang, B., and Chen, T., "Task migration enabling grid workflow application rescheduling," in [*Asia-Pacific Web Conference*], 130–135, Springer (2008).

[42] Swiecicka, A., Seredynski, F., and Zomaya, A. Y., "Multiprocessor scheduling and rescheduling with use of cellular automata and artificial immune system support," *IEEE Transactions on Parallel and Distributed Systems* **17**(3), 253–262 (2006).

[43] Deelman, E., Blythe, J., Gil, Y., and Kesselman, C., "Workflow management in griphyn," in [*Grid Resource Management*], 99–116, Springer (2004).

[44] Kao, Y.-H., Wright, K., Huang, P.-H., Krishnamachari, B., and Bai, F., "Mabsta: Collaborative computing over heterogeneous devices in dynamic environments," in [*IEEE INFOCOM 2020-IEEE Conference on Computer Communications*], 169–178, IEEE (2020).

[45] Satyanarayanan, M., "Cloudlets: at the leading edge of cloud-mobile convergence," in [*Proceedings of the 9th international ACM Sigsoft conference on Quality of software architectures*], 1–2 (2013).

[46] Chen, M., Hao, Y., Lai, C.-F., Wu, D., Li, Y., and Hwang, K., "Opportunistic task scheduling over co-located clouds in mobile environment," *IEEE Transactions on Services Computing* **11**(3), 549–561 (2016).

[47] Rahimi, M. R., Venkatasubramanian, N., Mehrotra, S., and Vasilakos, A. V., "Mapcloud: Mobile applications on an elastic and scalable 2-tier cloud architecture," in [*2012 IEEE Fifth International Conference on Utility and Cloud Computing*], 83–90, IEEE (2012).

[48] Ra, M.-R., Sheth, A., Mummert, L., Pillai, P., Wetherall, D., and Govindan, R., "Odessa: enabling interactive perception applications on mobile devices," in [*Proceedings of the 9th international conference on Mobile systems, applications, and services*], 43–56 (2011).

[49] Ramachandran, G. S., Garcia, L. A., and Krishnamachari, B., "Distributed computing for internet of things under adversarial environments," *IoT for Defense and National Security, Eds. Robert Douglass, Keith Gremban, Ananthram Swami, and Stephan Gerali, Wiley (book to appear)* .

[50] Farooq, M. J. and Zhu, Q., "On the secure and reconfigurable multi-layer network design for critical information dissemination in the internet of battlefield things (iobt)," *IEEE Transactions on Wireless Communications* **17**(4), 2618–2632 (2018).

[51] Williams, R. K., Gasparri, A., and Krishnamachari, B., "Route swarm: Wireless network optimization through mobility," in [*2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*], 3775–3781, IEEE (2014).

[52] Farooq, M. J. and Zhu, Q., "A multi-layer feedback system approach to resilient connectivity of remotely deployed mobile internet of things," *IEEE Transactions on Cognitive Communications and Networking* **4**(2), 422–432 (2018).