A Novel Task Offloading Scheme for Robotics Applications in Information Centric Networks

Daniele Sparapano*, Francesco Vista*†, Paolo Benedetti*, Giuseppe Piro*† and Luigi Alfredo Grieco*†
*Dept. of Electrical and Information Engineering, Politecnico di Bari, Italy,
†CNIT, Consorzio Nazionale Interuniversitario per le Telecomunicazioni, Italy,
email: {name.surname}@poliba.it

***Abstract*—This contribution proposes a novel task offloading scheme for robotics applications in Information-Centric Networks. The reference scenario includes heterogeneous agents connected to various wireless Network Attachment Points, asking for computational services to Multi-access Edge Computing servers deployed at the network edge. It also uses Information-Centric Networking facilities to implement resource announcement, resource discovery, and service provisioning. The effectiveness of the conceived solution has been tested through experimental tests in realistic scenarios.**

***Index Terms*—Information Centric Networking, Industry 4.0, Task offloading.**

I. Introduction

In the Industry 4.0 domain, the Industrial IoT (IIoT) paradigm is supporting the definition of new use cases requiring low latency and high-reliability constraints. Considering that typical IIoT applications such as Autonomous Robotics, Additive Manufacturing, Digital Manufacturing, and Mixed Reality (MR) involve different devices with limited computational and energy capabilities [1], complex tasks can be only provided by demanding computational activities to remote nodes. As a result, devolving computational activities to the cloud, fog, and edge computing is becoming more common. In the context of fifth-generation (5G) and Beyond 5G (B5G) communication systems, differently from conventional approaches, Multi-access Edge Computing (MEC) can be used to guarantee heavy computational capabilities close to the end-users, as well as ensure lower and predictable communication latency in the related service provisioning [2], [3], [4]. Indeed, MEC represents one of the most important key enabling technology for the IIoT.

Despite the advantages, new challenges emerge. In fact, during the task offloading procedure, it is fundamental (and, depending on the target use case, also challenging) to dynamically manage the allocation of services' requests at the network edge while considering the upper bound capabilities of the available MEC servers. This issue has been extensively investigated in conventional IP-based strategies (e.g., through centralized or decentralized strategies [5]-[12]), but it remains far less explored in Future Internet architectures based on the Information-Centric Network (ICN) paradigm. As well known, ICN represents a different communication paradigm able to solve typical issues affecting IP-based deployments, while providing important performance gains in terms of security, bandwidth consumption, latency, and energy consumption in data dissemination services [13]-[17], also in wireless mesh network [18] and in the IIoT application domains [19], [20].

At the time of this writing, and to the best of the authors' knowledge, task offloading problems in ICN networks have been initially investigated in [21]-[25]. Despite the interesting solutions presented in these works, there are still some open issues deserving more attention, such as: i) some strategies require an extension to the native protocol architecture [21], ii) other contributions do not explicitly consider the actual capabilities of MEC servers [22], [23], iii) sometimes the allocation of users' requests is done via an optimization algorithm that increases complexity and reduces scalability [24], and iv) some approaches may lose their effectiveness when used on a large scale [25].

To bridge this gap, this work proposes a novel task offloading scheme for ICN networks. Without loss of generality, it focuses on robotics applications and considers a reference scenario where: i) heterogeneous agents are connected to various wireless Network Attachment Points (NAPs), ii) NAPs are connected to each other through a wired network grid, and iii) MEC servers are deployed at the network edge to provide task offloading functionalities. More specifically, the proposed approach introduces the following main contributions. First, a custom namespace has been designed to describe services' requests and available computational capabilities at the network edge. Second, ICN facilities have been used to manage resource announcement, resource discovery, and service provisioning. Indeed, the resulting solution appears fully data-centric, efficient, scalable, and distributed. The effectiveness of the conceived solution has been tested through experimental tests in realistic scenarios embracing a mix of MR, Automated Guided Vehicles (AGVs), Unmanned Aircraft System (UAS), and IIoT devices.

The rest of the paper is organized as follows. Section II provides a summary of the current state of the art addressing the task offloading strategies in ICN networks. Section III describes the reference scenario taken into account in this contribution and illustrates the proposed methodology. Section IV presents a performance assessment of the proposed methodology, carried out through experimental tests in realistic scenarios. Finally, Section V concludes the work and clarifies future research directions.

II. State of the art on task offloading strategies in ICN networks

In conventional IP-based network architectures, task offloading can be managed via centralized and distributed techniques, as explicitly reported in a recent survey [5]. For instance, one or more controllers can be configured to distribute requested jobs among the available MEC servers, according to application type, latency requirements, and resource types required [6], i.e., by exploiting classical optimization techniques [7][8], Machine Learning (ML) algorithms [9][10] or deep reinforcement learning approaches [11][12]. At the same time, ICN has been identified as a valid and effective alternative to sustain the definition of the Future Internet [13][14], thanks to its ability to optimize data retrieval and in-caching mechanisms [15], improve the Quality of Experience in multimedia streaming services [16], and support consumer mobility [17]. The potentials of ICN have been studied also in the Internet of Things (IoT) context [19][20]. Moreover, at the time of this writing, and to the best of the authors' knowledge, only a few contributions explored the task offloading problem in ICN networks, mainly implemented through the Named Data Networking (NDN) paradigm.

For instance, [21] proposes an extension of the baseline NDN architecture, namely Name Data Networking at the edge (NDNe), supporting task offloading. Here, the consumer issues a service request through an enhanced INTEREST packet, carrying the service name and the related requirements (e.g., maximum delay and energy consumption for executing the task). By gradually increasing its Time-To-Live (TTL) value, such a request iteratively propagates across the network until nodes capable of providing the requested service (simply referred to as service providers) are reached. Indeed, the suitable service provider replies with an enhanced DATA packet, including the estimated time and energy consumption needed to execute the task. After that, the consumer selects the best service provider and the task offloading operation starts. The approach is very interesting, but it modifies the native NDN protocol and does not use ICN network facilities to autonomously manage service discovery and task allocation.

The work presented in [22] presents a strategy to execute specific functions at the network edge. These functions are identified by names, downloaded locally into edge nodes (according to a score function), and executed on a lightweight Virtual Machine. This solution is related to task offloading, but it does not take care of the actual computational availability of edge nodes. At the same time, it prioritizes popular functions while determining an increment in the response latency for services that experience low score values.

Another framework to dynamically distribute IoT data processing at the network edge is presented in [23]. It uses a custom naming scheme for identifying IoT content and services. However, they extend the NDN architecture with a data structure to take into account the available service and, similar to [22], it is updated according to the request popularity expressed through a score function. Thus, also in this case, the resulting approach does not consider the resource availability of edge nodes.

The solution presented in [24] adopts ICN functionalities to dynamically announce resource availability and distribute users' requests. Here, however, the selection of the MEC server to be involved in task offloading operations is delegated to the routers and solved through a multi-objective optimization problem. The resulting approach requires that routers must explicitly know the available MEC servers and actively participate in the task offloading procedure. This may increase complexity and reduce scalability in complex deployments.

In [25], instead, a strategy to distribute the status of available resources is addressed by employing scoped-flooding. Specifically, edge nodes distribute Resource Bread-Crumbs (RBCs), which contain information on the available resources and include other metrics, such as the number of hops and routing cost. However, when available resources change, new RBCs are distributed only within edge node proximity (i.e., until the metrics in the RBCs exceed the threshold). Accordingly, the solution seems to lose its effectiveness in large-scale deployments.

In conclusion, the study of the state of the art highlights that solutions supporting task offloading in ICN networks still present some limitations. Therefore, to enrich the scientific literature in this context, this paper proposes a novel strategy that addresses the task offloading problem in order to: i) avoid to modify NDN functionalities (differently from [21]), ii) announce computational capabilities by always considering the actual MEC capabilities (differently from [22], [23]), iii) distribute users' requests via pure NDN facilities and without requiring the implementation of complex tasks for network nodes (differently from [24]), iv) easily support task offloading in large scale scenarios (differently from [25]).

III. The proposed solution

Fig. 1 shows the IIoT network architecture considered in this work. Here, several NAPs offer wireless connectivity (e.g., Wi-Fi, 4G, and 5G) to groups of heterogeneous agents distributed within an industrial environment. Each NAP covers a limited geographical area, namely femtocell, and it is connected to the others through a cabled or wireless network grid. Hence, at the network edge, beyond each NAP, MEC servers with a co-located NDN router are deployed to offer task offloading to IIoT agents. Based on these premises, this work assumes that these agents, attached to different NAPs, desire to offload some tasks to the network edge, i.e., by exploiting the computational resources exposed by MEC servers.

The task offloading problems can be addressed by fully leveraging ICN functionalities (and specifically the NDN protocol [26]), also in wireless mesh network. The resulting approach is scalable, flexible, and fully distributed. The following sections will deeply describe how computational resources are i) identified through custom content names and ii) announced and discovered through NDN primitives.

TABLE I
EXAMPLE OF NAMESPACE

Service Type	CPU (MI)	Memory (GB)	Storage (GB)	Service Category	Namespace
MR	200	1.3	5	CAT 1	n2n://industry/mr/cat1
MR	400	1.3	5	CAT 2	n2n://industry/mr/cat2
AGV	50	2	1	CAT 1	n2n://industry/agv/cat1
AGV	150	2	1	CAT 2	n2n://industry/agv/cat2
IIoT	30	0.5	10	CAT 1	n2n://industry/iiot/cat1
IIoT	50	0.5	10	CAT 2	n2n://industry/iiot/cat2

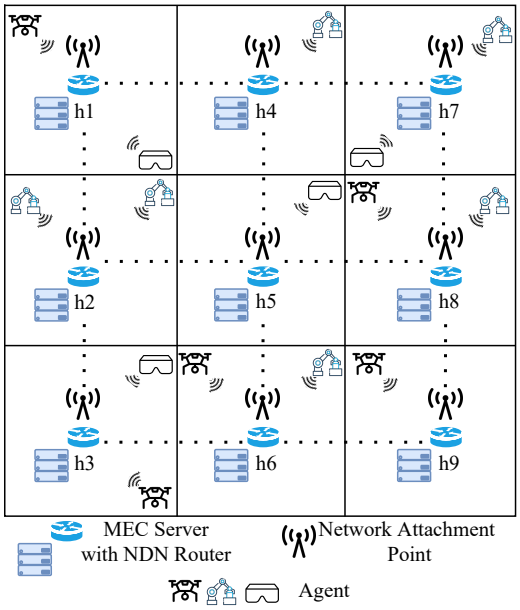


Fig. 1. The reference network architecture.

A. Namespace Design

The namespace has been properly designed in order to effectively and dynamically announce the ability of MEC servers to support specific services according to their available computational capabilities. Indeed, that design carefully considers typical services required in the industrial environment, such as MR devices, AGVs, UAS, and IIoT devices, and the different computational capabilities they may request. Some significant examples, according to the current literature [27] are reported in Table I. The resulting namespace is also effective to support resource discovery and provisioning.

In the conceived approach, each MEC server is characterized by its computational capabilities in terms of CPU, RAM memory, and storage. These capabilities can be used to serve agents' requests belonging to a given *Service Type* and *Service Category*. The *Service Type* explicitly identifies the type of agent within the network. The *Service Category*, instead, reports the upper bound of computational capabilities that the IIoT agent may request for a specific task, expressed in terms of: (i) CPU, expressed in Million Instructions Per Seconds (MIPS) units, represents the maximum number of MIPS an agent may require from a MEC server, (ii) RAM memory, expressed in GB, indicates the maximum RAM memory occupancy on the MEC server, and (iii) storage, expressed in GB, denotes the maximum amount of storage memory a task needs.

As shown in Table I, the proposed approach adopts a namespace structure like:

n2n://industry/[service_type]/[service_category].

B. Resource Announcement

MEC servers continuously monitor their available computational resources. When needed, they implement the resource announcement procedure, willing to clarify the list of services they are able to support and, accordingly, update their routers' Link State Database (LSDB).

To define which names can be announced, the MEC server implements an iterative procedure. In particular, the MEC server holds the list of available services defined in the reference scenario. The i-th service may request up to c_i of CPU, m_i of RAM memory, and s_i of storage memory. Therefore, for each service, the MEC server verifies if it has enough computational resources. In the affirmative case, it announces $name_i$, which includes the service type and service category, through its router.

Then, announced names propagate through the network by following the typical name distribution mechanism conceived for Named-data Link State Routing (NLSR) networks [28]. Indeed, each router in the network periodically checks if the other routers have new available content names and eventually triggers synchronization mechanisms to update its LSDB. A change in the LSDB state is propagated through Link State Advertisements (LSA) message to nearby routers. Upon receiving the new LSA, the routers update their LSDB, recalculate the paths, and update their Forwarding Information Bases (FIBs).

In particular, each router sends a Sync INTEREST message to the others, entailing a hash of all the content names registered locally with NLSR and those injected by connected routers. When a router receives a Sync INTEREST, it compares the hash in the content name with the hash generated by its LSDB. If the hashes mismatch, the router answers with a Sync Reply message enclosing the missing LSA name. Upon receiving the Sync Reply message, the router with a missing LSA sends a LSA INTEREST message to retrieve it. Finally, the router answers with a LSA DATA packet enclosing the new LSA.

C. Resource discovery and service provisioning

The resource discovery procedure allows agents to find a MEC server in the network able to perform task offloading.

As a first step, the agent sends a task offloading request to the reference NAP, by using the namespace structure described in Section III-A. Then, the NAP routes the request through the network until an NDN router, which announces the required service, is reached. Finally, it replies by performing a subscription to the right agent in order to execute task offloading operations.

Fig. 2 shows in detail the message exchange during the proposed resource discovery procedure. Specifically, an agent sends a resource request through an INTEREST message (Step 1). Once the INTEREST message reaches the first NDN router with the requested service, it replies to the agent through a DATA message, signaling the availability of its MEC server to execute task offloading operations (Step 2). Thus, the MEC server reserves the computation resources to the agent for a given time. Furthermore, it checks the current amount of available resources and, if necessary, its NDN router broadcasts a new adjacency LSA to the entire network. The most recent version of the LSAs is saved in the LSDB, in accordance with the NLSR protocol [28]. Then, the designated NDN router sends a semi-permanent INTEREST to subscribe to the agent (Step 3), by following the same approach stated in [29], [17]. This enables the agent to send DATA messages (with input data), through the network, to the MEC server (Step 4) offering task offloading services.

It is worth noting that, in the conceived approach, the DATA messages of NDN routers require freshness to avoid conflicts over name requests. Hence, they are not stored within the Content Store (CS) of intermediate nodes. Finally, at the end of the task offloading operations, the resources will again be available within the MEC server, so the available services must be updated via resource announcement procedure.

D. Running example

For the sake of clarity, this section provides a running example further explaining the workflow that characterizes the proposed solution. The considered scenario is composed of a MR agent and 3 NDN routers, each one linked to the corresponding MEC servers. Moreover, as depicted in Fig. 3, each MEC server announces the service it can offer among those listed in Tab. I through its co-located NDN router.

Figure 3a shows that the MR agent issues a request for the service `n2n://industry/mr/cat1`, through an INTEREST packet. Since the MEC server attached to the router h1 cannot support the requested service, the INTEREST message is delivered to h2 (according to its FIB). Then, the node h2 replies to the agent with an ACK message, followed by a semi-permanent INTEREST message. Finally, the h1's Pending Interest Table (PIT) table is updated with the information of the requesting agent for routing subsequent messages (see Figure 3b).

IV. Performance Assessment

The performance of the proposed task offloading scheme in realistic deployments is investigated in this paper through experimental tests. To this end, the Mini-NDN emulation tool

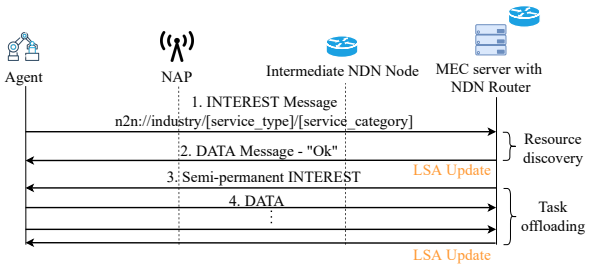
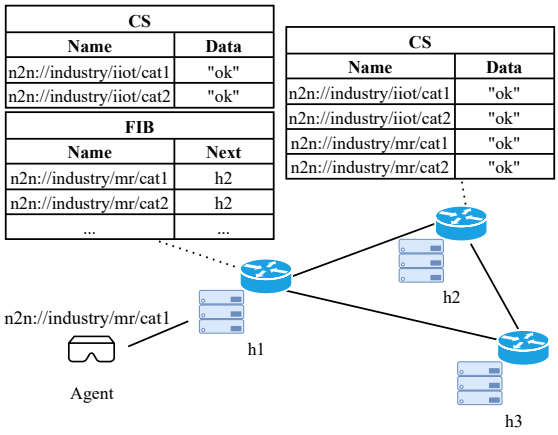
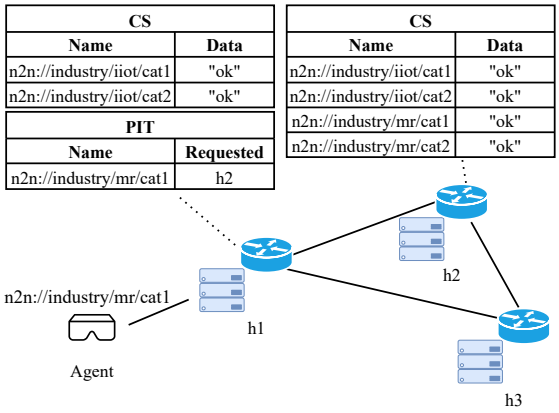


Fig. 2. Messages exchanged during the service provisioning.



(a) Before the agent request



(b) After the agent request

Fig. 3. Running example.

is used to create a network architecture, as depicted in Fig. 1, composed of a grid of 3x3 NDN routers with their MEC servers and NAPs, as well as all the functionalities described in Section III. Specifically, each MEC server is configured with the following parameters: RAM = 16 GB, storage = 240 GB, and CPU = 10^5 MIPS, according to the industrial computer state of the art [30].

To demonstrate the proper behavior of the proposed solution, this work evaluates the occupancy of MEC servers (i.e., CPU, RAM, and storage memory) while agents' requests are satisfied by considering two representative scenarios. The first assumes that agents are directly connected to a single NAP,

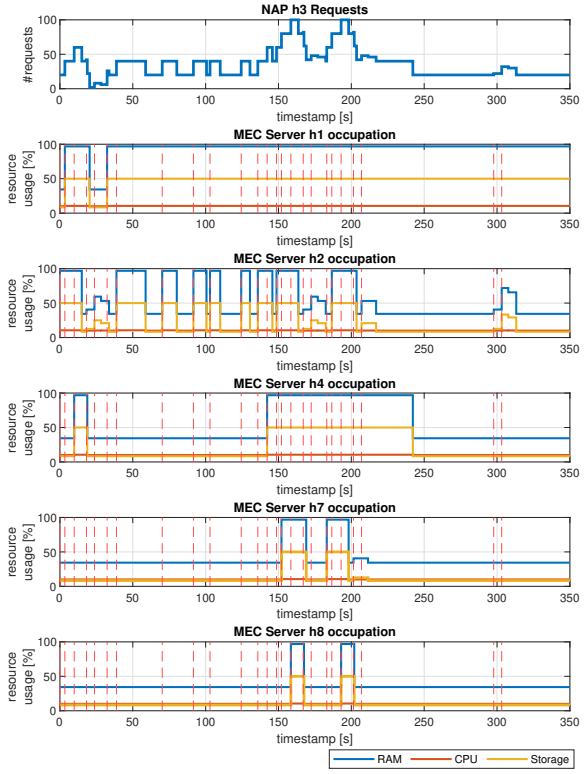


Fig. 4. Proof of operation of the proposed algorithm with agents connected to a single NAP.

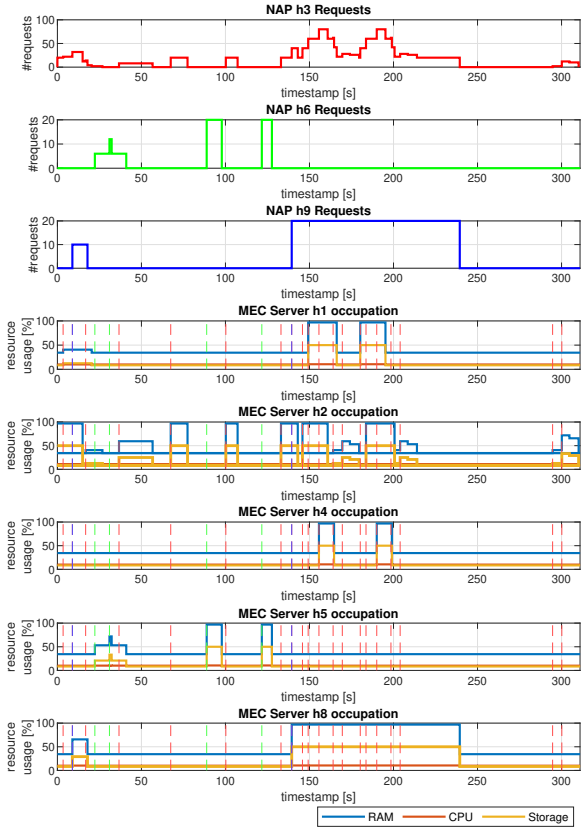


Fig. 5. Proof of operation of the proposed algorithm with agents connected to multiple NAP.

whereas the second considers that agents' requests are sent to multiple NAPs. Moreover, in both scenarios, MEC servers h3, h6, and h9 are enforced to be out of service, thus they cannot serve any agents' requests.

It is worth noting that the communication technology employed between the agent and NAPs does not affect the effectiveness of proposed solution.

A. *Agents connected to a single femtocell*

The first scenario considers different IIoT agents directly connected to the NAP h3.

The experimental results, depicted in Fig. 4, show the total number of agents' requests and the resource occupancy of MEC servers over time. In particular, in the beginning, it can be observed that agents send a total of 20 requests to the NAP h3. These requests are initially served by the MEC server h2, filling its RAM. As expected, MEC server h2 stops exposing the service name to avoid resource overloading. As a result, the following agents' requests are sent to the MEC server h1 (through h2, which acts as a router). When node h2 completes its assigned operations, it exposes services again and resources are allocated for other requests. Furthermore, as the number of requests increases, i.e., between 150 s and 200 s, MEC servers h7 and h8 are involved in the task offloading process as well. This is because the nearest MEC servers have already allocated all of their computational resources. At the end of task offloading operations, connections between MEC servers and agents are closed, and then the allocated computing resources become available again.

Results demonstrate that the proposed methodology allows to offload tasks efficiently and distribute them over the network by always (i.e., in real-time) considering the MEC servers' occupancy.

B. *Agents connected to multiple femtocells*

The second scenario, instead, considers different IIoT agents connected to the NAPs h3, h6, and h9.

Experimental results, depicted in Fig. 5, show that the first requests received by NAP h3, h6, and h9 are fulfilled by MEC servers h2, h5, and h8, respectively. The subsequent requests, instead, are routed and then fulfilled by the remaining MEC servers in the network. In particular, when the number of requests to the NAP h3 grows, i.e., between 150 s and 200 s, they are initially satisfied by MEC server h1 since h2 is not available. Then, when the resources of MEC servers h2 become available again, it may serve the following requests.

This shows that the proposed approach allows to i) fulfill the requests of the agents by the MEC servers closest to their reference NAPs and ii) reduce network traffic and evenly distribute service load by routing the agents' requests according to the rules defined within the FIB tables.

V. Conclusions

This work proposed a novel task offloading scheme by fully leveraging ICN functionalities. The reference scenario involves heterogeneous agents connected to different wireless NAPs and requesting computational services from MEC servers placed at the network edge. A custom namespace has been designed to describe the service request and the available computational capabilities provided by edge nodes. Furthermore, the NDN protocol has been used to manage resource announcements and discovery, as well as service provisioning. An experimental campaign has been carried out to prove the benefits of the proposed strategy. The obtained results show that the proposed approach efficiently offloads tasks and distributes them over the network while accounting for the MEC servers' occupancy. Moreover, future research will also take into account the agent delay requirements (e.g., by extending the service category fields) to select the best MEC server, as well as an extension of the naming scheme to support services with looser latency requirements and execute the demanded operations in the cloud to further spread the workload. At the same time, future work will investigate the performance of the proposed solution while accounting for the service latencies introduced by the communication technologies employed.

Acknowledgements

This work has been supported by the PRIN project no. 2017NS9FEY entitled "Realtime Control of 5G Wireless Networks: Taming the Complexity of Future Transmission and Computation Challenges" funded by the Italian MIUR, and by the project entitled "The house of emerging technologies of Matera (CTEMT)" funded by the Italian MISE. It has been also supported by the Italian MIUR PON projects AGREED (ARS01 00254).

References

[1] K. Tange, M. De Donno, X. Fafoutis, and N. Dragoni, "A Systematic Survey of Industrial Internet of Things Security: Requirements and Fog Computing Opportunities," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, 2020.

[2] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge Computing in Industrial Internet of Things: Architecture, Advances and Challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, 2020.

[3] B. Liang, M. A. Gregory, and S. Li, "Multi-access Edge Computing Fundamentals, Services, Enablers and Challenges: A Complete Survey," *Journal of Network and Computer Applications*, vol. 199, 2022.

[4] G. Nain, K. Pattanaik, and G. Sharma, "Towards edge computing in intelligent manufacturing: Past, present and future," *Journal of Manufacturing Systems*, vol. 62, 2022.

[5] L. Huang, X. Feng, C. Zhang, L. Qian, and Y. Wu, "Deep reinforcement learning-based joint task offloading and bandwidth allocation for multi-user mobile edge computing," *Digital Communications and Networks*, vol. 5, no. 1, 2019.

[6] A. Islam, A. Debnath, M. Ghose, and S. Chakraborty, "A Survey on Task Offloading in Multi-access Edge Computing," *Journal of Systems Architecture*, vol. 118, 2021.

[7] W. Ni, H. Tian, X. Lyu, and S. Fan, "Service-dependent task offloading for multiuser mobile edge computing system," *Electronics Letters*, vol. 55, no. 15, 2019.

[8] F. Wei, S. Chen, and W. Zou, "A Greedy Algorithm for Task Offloading in Mobile Edge Computing System," *China Communications*, vol. 15, no. 11, 2018.

[9] J. Wang, J. Hu, G. Min, A. Y. Zomaya, and N. Georgalas, "Fast Adaptive Task Offloading in Edge Computing Based on Meta Reinforcement Learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, 2020.

[10] T. Alfakih, M. M. Hassan, A. Gumaei, C. Savaglio, and G. Fortino, "Task Offloading and Resource Allocation for Mobile Edge Computing by Deep Reinforcement Learning Based on SARSA," *IEEE Access*, vol. 8, 2020.

[11] Y. Dai, K. Zhang, S. Maharjan, and Y. Zhang, "Deep Reinforcement Learning for Stochastic Computation Offloading in Digital Twin Networks," *IEEE Trans. Industr. Inform.*, vol. 17, no. 7, 2021.

[12] Y. Ren, Y. Sun, and M. Peng, "Deep Reinforcement Learning Based Computation Offloading in Fog Enabled Industrial Internet of Things," *IEEE Trans. Industr. Inform.*, vol. 17, no. 7, 2021.

[13] D. Gupta, S. Rani, S. H. Ahmed, S. Verma, M. F. Ijaz, and J. Shafi, "Edge Caching Based on Collaborative Filtering for Heterogeneous ICN-IoT Applications," *Sensors*, vol. 21, no. 16, 2021.

[14] N. Dinh and Y. Kim, "An Energy Reward-Based Caching Mechanism for Information-Centric Internet of Things," *Sensors*, vol. 22, no. 3, 2022.

[15] K. Yu, S. Eum, T. Kurita, Q. Hua, T. Sato, H. Nakazato, T. Asami, and V. P. Kafle, "Information-Centric Networking: Research and Standardization Status," *IEEE Access*, vol. 7, 2019.

[16] A. A. Barakabitze, N. Barman, A. Ahmad, S. Zadtootaghaj, L. Sun, M. G. Martini, and L. Atzori, "QoE Management of Multimedia Streaming Services in Future Networks: A Tutorial and Survey," *IEEE Commun. Surveys Tuts*, vol. 22, no. 1, 2020.

[17] P. Benedetti, G. Piro, and L. A. Grieco, "A softwarized and MEC-enabled protocol architecture supporting consumer mobility in Information-Centric Networks," *Computer Networks*, vol. 188, 2021.

[18] S. Kalafatidis, V. Demiroglou, L. Mamatas, and V. Tsaoussidis, "Experimenting with an SDN-Based NDN Deployment over Wireless Mesh Networks," *Proc. of IEEE Conf. on Computer Commun. Workshops (INFOCOM wkshps)*, 2022.

[19] R. Sebastian, G. Giambene, and T. d. Cola, "Information-centric networking application to maritime satellite communications," in *Proc. of IEEE Int. Conf. on Communications (ICC)*, 2020.

[20] A. Sethi, S. Vijay, and V. Gupta, "Mobility and QoS based cross network for ICN Vehicular Communication," *Procedia Computer Science*, vol. 171, 2020.

[21] M. Amadeo, C. Campolo, and A. Molinaro, "NDNe: Enhancing Named Data Networking to Support Cloudification at the Edge," *IEEE Commun. Lett.*, vol. 20, no. 11, 2016.

[22] M. Król and I. Psaras, "NFaaS: Named Function as a Service," in *Proc. of the 4th ACM Conf. on Information-Centric Networking*, ser. ICN '17. New York, NY, USA: Association for Computing Machinery, 2017.

[23] M. Amadeo, C. Campolo, A. Molinaro, and G. Ruggeri, "IoT Data Processing at the Edge with Named Data Networking," in *Proc. of European Wireless Conference*. VDE, 2018.

[24] S. Mastorakis, A. Mtibaa, J. Lee, and S. Misra, "ICedge: When Edge Computing Meets Information-Centric Networking," *IEEE Internet Things J.*, vol. 7, no. 5, 2020.

[25] D. Kondo, T. Ansquer, Y. Tanigawa, and H. Tode, "Resource Discovery for Edge Computing over Named Data Networking," in *Proc. of IEEE Annual Computers, Software, and Applications Conf. (COMPSAC)*, 2021.

[26] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A Survey of Information-Centric Networking Research," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 2, 2013.

[27] S. Massari, N. Mirizzi, G. Piro, and G. Boggia, "An Open-Source Tool Modeling the ETSI-MEC Architecture in the Industry 4.0 Context," in *Proc. of Mediterranean Conf. on Control and Automation (MED)*, 2021.

[28] L. Wang, V. Lehman, A. K. M. Mahmudul Hoque, B. Zhang, Y. Yu, and L. Zhang, "A Secure Link State Routing Protocol for NDN," *IEEE Access*, vol. 6, 2018.

[29] A. V. Ventrella, L. A. Grieco, and G. Piro, "Information-Centric Networking in Environmental Monitoring: an overview on publish-subscribe implementations," in *Proc. of IEEE Int. Conf. on Advanced Video and Signal Based Surveillance (AVSS)*, 2017.

[30] Fujitsu. (2022) Fujitsu iot solution intelliedge. [Online]. Available: https://www.fujitsu.com/emeia/products/computing/pc/edge-computing