# Towards Named AI Networking: Unveiling the Potential of NDN for Edge AI

Claudia Campolo[1], Gianmarco Lia[1], Marica Amadeo[1], Giuseppe Ruggeri[1], Antonio Iera[2], and Antonella Molinaro[1]

[1] University Mediterranea of Reggio Calabria, Italy name.surname@unirc.it
[2] University of Calabria, Italy
antonio.iera@dimes.unical.it

**Abstract.** Thanks to recent advancements in edge computing, the traditional centralized cloud-based approach to deploy Artificial Intelligence (AI) techniques will be soon replaced or complemented by the so-called *edge AI* approach. By pushing AI at the network edge, close to the large amount of raw input data, the traffic traversing the core network as well as the inference latency can be reduced. Despite such neat benefits, the actual deployment of edge AI across distributed nodes raises novel challenges to be addressed, such as the need to enforce proper addressing and discovery procedures, to identify AI components, and to chain them in an interoperable manner. Named Data Networking (NDN) has been recently argued as one of the main enablers of network and computing convergence, which edge AI should build upon. However, the peculiarities of such a new paradigm entails to go a step further. In this paper we *disclose the potential of NDN to support the orchestration of edge AI*. Several motivations are discussed, as well as the challenges which serve as guidelines for progress beyond the state of the art in this topic.

**Keywords:** Edge computing · Information Centric Networking · Named Data Networking · Internet of Things · Artificial Intelligence · Edge AI.

## 1 Introduction

The research interest in Artificial Intelligence (AI) was recently boosted by the advancements in cloud computing and the massive deployment of Internet of Things (IoT) devices. Indeed, several IoT applications, such as video surveillance, autonomous driving, smart home appliance and industrial automation, greatly benefit from the use of AI capabilities, including data, image, audio, and video analysis. Among AI algorithms, Deep Learning (DL) methods consist of two phases: *training phase* and *inference phase*. The first one has the purpose to set, according to input data, the weights of the Artificial Neural Network (ANN) by which, during the second phase, decisions (e.g., classification, recognition) are taken. Such operations are memory- and power-hungry. Hence, typically, resource-constrained IoT devices just send the data streams they collect/sense to the remote cloud. Mega-scale data-centers, with their virtually


DOI https://doi.org/10.1007/978-3-030-61746-2_2

unlimited capabilities, are then responsible for processing raw data and deriving knowledge to be sent back to requesting devices/users. The emergence of the edge computing paradigm, by bringing computing resources closer to devices, paves the way for re-engineering the way in which AI solutions are deployed. If DL services are deployed at the edge, close to where input data are produced and likely consumed, the latency and cost of sending data to the cloud for model training/inference will be reduced, while also offloading the core network [1].

Following the groundbreaking paradigm pushing AI to the edge, referred to as *edge AI*, the AI model training and inference tasks can be performed across several edge nodes, such as base stations, backhaul nodes, and IoT gateways. The cloud data center capabilities are used whenever additional processing power is needed and trained models are to be stored.

Despite the numerous literature works targeting the orchestration of edge computing resources to tackle the placement of generic computing tasks, several peculiar issues arise when dealing with the orchestration of AI workloads. First, placement decisions should be taken not only to ensure that data collection latencies, computation times and/or energy consumption are minimized, but also to provide the needed inference accuracy of trained ANNs. Heterogeneous devices may provide inference results with different accuracy levels, e.g., according to their computing capabilities. Second, similarly to contents, AI inference results, once computed, could serve different requests. Finally, another issue for distributed AI at the edge, unlike in centralized AI deployments, is the lack of interoperability due to fragmented and mainly application-specific solutions [2]. Hence, it is crucial to identify and discover AI components to build intelligent applications upon them, while efficiently using network and computing resources.

In this paper we promote the usage of the Named Data Networking (NDN) paradigm (https://named-data.net/) to deal with the aforementioned issues. Originally conceived as an innovative content delivery solution, NDN has been more recently overhauled to deal with edge computing [3–6], as scanned in Section 2. Its native in-network caching capability coupled with the semantic-rich naming scheme would play a crucial role in facilitating the deployment of edge AI solutions, as we discuss in Section 3. To the best of our knowledge, this is the first work that discloses the potential of NDN for orchestrating edge AI.

## 2 From named contents to named services

NDN [7] implements name-based consumer-driven communication based on the exchange of two packets types, *Interest* and *Data*, originally used to request and provide authenticated contents, respectively. One Data packet exactly consumes one Interest packet. Both packets carry hierarchical Uniform Resource Identifier (URI)-like names which uniquely identify the content. There are no specific restrictions in the way name components can be defined. Data packets also piggyback the publisher's signature and other authentication information to enable per-packet security. By design, NDN provides in-network caching and multicast support. Each node maintains a Content Store (CS) to cache incoming Data

packets according to local storage policies. This allows intermediate nodes to satisfy future requests and speed up data retrieval, while reducing the amount of traffic crossing the network. A Pending Interest Table (PIT) and a Forwarding Information Base (FIB) are maintained to, respectively, record the pending requests that wait to be consumed by Data packets and identify the outgoing interfaces (and relevant attributes, e.g., latency) to forward the Interests.

Recently, the NDN logic has been extended to complement data retrieval by data processing. According to the pioneering vision in [8], referred to as Named Function Networking (NFN), an NDN name can be used to identify contents, processing functions and/or a combination of them. A consumer can request a named function to be applied over a content (e.g., video compression), and the network uses advanced routing-by-name mechanisms to discover both the content and the node in charge of executing that function and returning the processed content. This output can be cached to satisfy future requests without the need of performing the computation again. NFN was extended in [3] to tackle wireless edge domains, where mobile consumers broadcast requests to offload a computing service to a more powerful node in the neighbourhood. The Interest packet is extended to carry the name of the processing function and a set of attributes that describe the consumer's demands, e.g., maximum tolerated latency, to allow a potential provider to self-candidate for the task execution. A smart deferral scheme is defined that lets the best provider (i.e., the one that executes the service in the shortest time) answer first. Therefore, on receiving the first response, the consumer immediately offloads the task to that node. The work in [5] identifies two distinct mechanisms to enable computing services via NDN in a wired edge domain. In the proactive approach, edge nodes periodically advertise the functions they support and their resource utilization (e.g., CPU, storage) by piggybacking this information in the routing protocol messages. Vice versa, in the reactive approach, the information is transmitted at the reception of the service request from the consumer. In both cases, the node with the lowest resource utilization is selected as the executor in such a way as to guarantee the fair distribution of computation efforts. Conversely, in [6], an orchestration scheme is defined aiming at guaranteeing the lowest service execution time. It accounts for two terms: *(i)* the time needed to collect the data to be processed, which depends on the network status and the proximity to the data, and *(ii)* the processing time, which depends on the local available resources and the demands of the service. The selection of the executor is performed in a reactive way: at the Interest reception, each node computes its own service execution cost and the one with the lowest cost is selected as the executor. Different policies may be flexibly enforced by applying the same distributed orchestration logic.

## 3 Why NDN for edge AI?

In edge AI every edge node can contribute to the AI workflow by playing different roles. For instance, a node can provide the computing capabilities to run a specific ANN, although initially not locally available. Another one can own

the trained ANN model itself and perform the inference whenever requested and also cache it. Nodes can also contribute to partial local model training in the case of Federated Learning (FL) [9]. In such a challenging and dynamic context, NDN can play a crucial role to properly identify AI components, route requests towards the discovered ones, by accounting for their requirements, and also chain them. In the following, the motivations for the evolution of edge AI towards a *named AI networking* paradigm will be discussed, by also treasuring previous NDN literature and its consolidated extensions to support edge computing.

**Naming and discovery**. An addressing scheme is needed to identify all available AI components, i.e., models, model parameters, inference results, in the edge domain. This facilitates the discovery for subsequent composition of the intelligent service (set of services) exploiting the AI components. The semantic-rich NDN naming well suits the aforementioned need. Unique expressive NDN names can adequately identify input data and inference results as piece of contents. In addition, names can request the retrieval of a ANN model and/or its execution as well as the ANN weights. AI components can be definitely treated as first class citizens, similarly to contents in the vanilla NDN. They can be referred directly by their name, regardless of the identity of the node where they are actually hosted. For instance, the name */recognition/AlexNet/w* would allow a node to download the weights of the specific Convolutional Neural Network (CNN) model to be used for image recognition. The usage of well-known namespaces can facilitate AI interoperability, overstepping the current difficulties in letting fragmented AI applications and components interact.

**Caching**. AI inference results, once computed, can be reused and serve requests from different applications. For instance, a co-located group of tourists in a museum may need the same output from an object recognition module of an Augmented Reality (AR) application. Caching inference results can be highly helpful *(i)* to save valuable computing resources, *(ii)* to avoid the redundant input data exchange, and *(iii)* to reduce the inference latency [1]. The caching decision also applies to ANN models which a node can decide to keep locally. Implementing caching at the network layer, as foreseen by NDN, can be fast and flexible. Moreover, thanks to the Interest aggregation in the PIT, if multiple nodes need the same computation/model, only one request is forwarded towards the potential cacher, and the replies are sent in multicast saving network resources. Caching and replacement decisions can be taken by accounting for the popularity of the model/parameter/inference, but also for their temporal validity, not to waste the limited storage resources at the edge. Applications must have the capability to specify the *freshness* with which they want to receive a given inference result, similarly to what suggested in [10] for IoT contents.

**In-network processing**. Building upon the NDN/NFN philosophy, decisions about where to place AI components and how and which components to reach and chain (e.g., computational resources, cached inference results, ANN models, input data, weights) can be taken *in-network*. Specifically, an edge node can either reply to a request for a named inference result if it is eligible to satisfy it, or it can route the request towards a node which already stores the output,

to save processing resources. The latter case is depicted in Fig. 1. Instead, it may autonomously decide to execute it, if the latency to reach the cached inference (as tracked in the FIB) is higher than the one needed to retrieve the input data through Interest/Data packets exchange and locally perform the inference. Context-rich attributes can be used to specify the demands for AI components, e.g., the desired inference accuracy and latency, and help the decision about which components to reach and activate. An Interest with name */recognition/cars/Rome/intersection32/4June/11:12* coupled with an attribute specifying *acc >80* can be sent to request the recognition of cars passing by intersection 32 in the city of Rome on June 4th, between 11:00 and 12:00, with an accuracy higher than 80%. Only those nodes providing such feature could reply with Data packets. Attributes can also facilitate the discovery of clients (e.g., mobile devices) in charge of locally training a model in FL procedures. Such devices are typically randomly selected [9]. An NDN Interest-driven discovery solution like the one devised in [3] could make the selection smarter. An Interest can be sent which specifies the requested type of learning task and also related parameters through attributes. Then, candidate task executors reply according to their capabilities. Instead of receiving the first reply only, multiple replies can be collected with a single request, e.g., if the Long Lived Interest option is leveraged [11], to discover the best contributors.

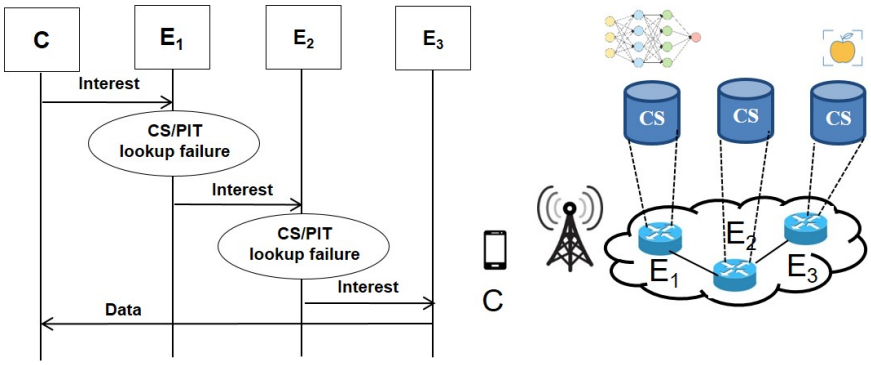


**Fig. 1.** Named AI networking workflow. $C$ requests an inference result; $E_1$ hosts the ANN model but it routes the request towards a cached inference result provided by $E_3$, which replies with a Data packet.

**Security**. Authentication for input contents as well as for models/inference to be transferred into Data packets is natively ensured in NDN. Indeed, such packets are digitally signed with a signature which is part of the Data Packet itself. Additional security countermeasures are needed; some of them have been investigated in the literature extending NDN for edge computing [6], [12], e.g., signed Interest packets, mutual consumer-provider authentication to verify the requester for an AI task is legitimate and the node providing the AI component is

**Table 1.** Matching between the main edge AI needs and NDN features.

| **Edge AI needs** | **NDN features** |
|---|---|
| Discovery and addressing of distributed and application-specific AI components | Well-known semantic-rich namespace |
| Reuse of inference results | In-network caching, request aggregation and multicast delivery |
| Placement, reachability and chaining of AI components | In-network processing, context-rich attributes, Long Lived Interests |
| Security | Data packet authentication, signed Interests |

authorized. On top of them, traditional solutions, like encryption and differential privacy mechanisms may be required for privacy preservation whenever sensitive data are exchanged for training/inference purposes [1].

## 4 Conclusions

In this paper we have proposed NDN, and its extensions, as an enabler of the transition from centralized AI to distributed edge AI, which well addresses its peculiarities (as summarized in Table 1). As a future work we plan to provide the algorithmic design of the NDN components and to quantitatively assess their benefits in orchestrating edge AI.

## 5 Acknowledgments

This work has been partially supported by the A COGnItive dynamic sysTem to allOw buildings to learn and adapt' (COGITO) project, funded by the Italian Government (PON ARS01_00836).

## References

1. X. Wang *et al.*, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.
2. E. Ramos *et al.*, "Distributing intelligence to the edge and beyond," *IEEE Computational Intelligence Magazine*, vol. 14, no. 4, pp. 65–92, 2019.
3. M. Amadeo *et al.*, "NDNe: Enhancing named data networking to support cloudification at the edge," *IEEE Comm. Letters*, vol. 20, no. 11, pp. 2264–2267, 2016.
4. C. Scherb *et al.*, "Resolution strategies for networking the IoT at the edge via named functions," in *2018 15th IEEE CCNC*, pp. 1–6.
5. A. Mtibaa *et al.*, "Towards edge computing over named data networking," in *2018 IEEE International Conference on Edge Computing (EDGE)*, pp. 117–120.
6. M. Amadeo *et al.*, "IoT services allocation at the edge via named data networking: From optimal bounds to practical design," *IEEE Trans. on Network and Service Management*, vol. 16, no. 2, pp. 661–674, 2019.

7. L. Zhang *et al.*, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.
8. C. Tschudin and M. Sifalakis, "Named functions and cached computations," in *IEEE CCNC*, 2014, pp. 851–857.
9. X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Network*, vol. 33, no. 5, pp. 156–165, 2019.
10. J. Quevedo *et al.*, "Consumer driven information freshness approach for content centric networking," in *2014 IEEE INFOCOM Workshop*, pp. 482–487.
11. A. Carzaniga *et al.*, "Content-based publish/subscribe networking and information-centric networking," in *ACM SIGCOMM workshop on Information-centric networking*, 2011, pp. 56–61.
12. M. Amadeo *et al.*, "Securing the mobile edge through named data networking," in *2018 IEEE WF-IoT*, pp. 80–85.