In-Network Dynamic Compute Orchestration Over Mobile Edge Systems

Roman Kovalchukov[1], Roman Glazkov[1], Srikathyayani Srikanteswara[2], Yi Zhang[2], Dmitri Moltchanov[1]
Gabriel Arrobo[2], Hao Feng[2], Marcin Spoczynski[2], Nageen Himayat[2]
[1]Unit of Electrical Engineering, Tampere University, Finland
[2] Intel Labs, USA
✉Contact author's e-mail: roman.glazkov@tuni.fi

***Abstract*—In this paper, we propose a new service orchestration approach for in-network fully distributed dynamic compute composition with limited involvement from the consumer and no centralized coordinator. The service is composed fully inside the network including assembling data and software, and compute node selection, leveraging standardized NDN functionalities. The use of the proposed approach will enable smooth support of dynamic compute services over wireless/mobile edge and edge/fog NDN-based systems, where the use of conventional IP approach faces fundamental difficulties related to dynamic allocation of IP addresses. We will also outline extensions of the proposed approach for streaming data, function chaining and re-use of partially computed results. Our numerical results show that the proposed method improves service reliability at the edge by increasing the Interest satisfaction by 5-10 times depending on the considered deployment, network topology, and resources.**

***Index Terms*—Dynamic edge computing; named data networking; in-network service orchestration; function chaining.**

I. Introduction

With the rapid development of Internet of Things (IoT) and Internet of Everything (IoE), the number of connected devices at the edge grows rapidly, leading to an increase in the amount of data being collected and processed. This gave rise to Edge and Fog Computing paradigms to bring the benefits and power of the cloud close to where data are created, analyzed, and acted upon. [1], [2].

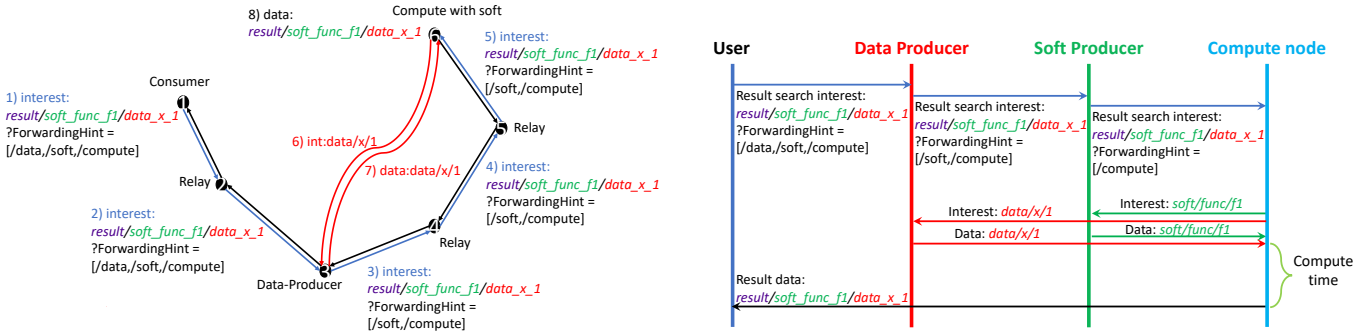
However, a large number of devices are also present beyond where traditional edge computing stops today, typically at the base station or within the operator's network, often referred to as a far edge. Examples of such far edge devices include laptops/desktops, tablets and smartphones, processors in vehicles or servers at businesses, etc. It is expected that the number of such devices [3] to be over 28 billion, making it a powerhouse of resources (sensors, storage, compute, data, etc.) at the far edge. They have the potential to form a horizontal layer of resources that can provide services to other clients. This aspect is today largely untapped since there is no easy way to harness these chaotic, potentially mobile resources that are not under a control of a single orchestrator and operate independently.

Named Data Networking (NDN) is a realization of Information Centric Networking (ICN) that can enable dynamic composition of compute services at the edge. NDN networks leverage flexible resources through a decentralized architecture, eliminating the need for a central entity to manage resource discovery and facilitate computational tasks on edge nodes. This approach has the potential for considerably enhancing user experience (e.g., via reduced latency) from composing/employing compute services as well as for facilitating new services, which are unavailable with the existing (IP-based) network architectures. While NDN was largely developed for data delivery over wired networks, its potential for enabling advanced services over the wireless network has not been deeply studied.

It has been demonstrated that NDN provides significant gains over conventional IP solutions and may efficiently operate in regimes, where IP solutions face difficulties, such as mesh disconnected from the Internet [4], [5]. However, the usage of NDN-based dynamic compute solution either presumes exact knowledge of the requested service or requires additional translation layer to be added to each compute node that maps the requested user function name to the set of available computes [6]. Both approaches are inherently centralized and do not account for network-wide knowledge of the service [7]. The reuse of computation has been shown to benefit the edge system because it can significantly speed up the execution of tasks at the edge [8], [9]. A framework described in [10] further enables computation reuse at the pervasive edge where a service may be offered by multiple distributed edge nodes by enhancing the naming design. Additional enhancements by efficient usage of the previously computed results which might be fully or partially available in the network were also combined in our work. Finally, there are value-added functionalities that can be added on top of it including the support for function chaining and streaming data. No such versatile solutions for dynamic compute orchestration has been proposed so far.

In this paper, we propose a dynamic compute service design for the mobile NDN-enabled edge systems. Instead of relying upon user to construct the service, we outsource this functionality to the network. By utilizing standardized functions of NDN networks, nodes coordinate themselves combining data, software and compute resource to execute the instructed task. The main contributions of the paper are:

- In-network dynamic compute service orchestration approach that does not require a central coordinator, relies



(a) Interest propagation in the network

(b) Signalling diagram

Fig. 1. Illustration of the "network makes decision" operation.

upon limited involvement from the consumer (user) in service provisioning, and allows for re-use of partially computed results minimizing redundant computations;

- Observation that outsourcing the compute service composition to the network addresses the problem of unreliable wireless access while NDN caching capabilities allow handling of extremely high load conditions;
- The comparison with a typical client-server approach reported in [11] showing that our method improves the Interest satisfaction by 5-10 times, depending on the system and deployment parameters.

Our paper is organized as follows. In Section II we review the related work. Further, in Section III we introduce the proposed solution. Enhancements are proposed in Section IV. We evaluate performance of the proposed solution in Section V. Conclusions are drawn in the last section.

II. Related Work

In [11], [12], the authors proposed two dynamic network orchestration mechanisms named "source makes decision" and "delegated node makes decision". The former approach assumes that the source is the one that not only initiates the service, but is also involved in the choice of the compute, software, and data resources that need to be utilized for service composition. In the delegated node solution, these functionalities are outsourced to one of the network nodes and the source is only responsible for initiating the service request. The main idea is to improve service reliability when running over unreliable wireless last mile access, e.g., Wi-Fi Networks. The biggest challenge of the "source makes decision" in [11] is the tight involvement of the consumer in the service orchestration, where compute server, data, and software selection need to be performed at the consumer. The "delegated node makes decision" alleviates these shortcomings but involves a third fully trusted party, that is responsible for service composition. This may require additional security measures and/or usage of specialized hardware/software at access points.

NFN (Named Function Networking) is an extension to existing ICN networks that enables an ICN network to deliver not only data that was already published but also on-demand results of computations, whereby the network can optimize where a computation is executed [13]. NFN performs in-network resolution of expressions instead of mere content retrieval of a single name. NFN orchestrates computations for a specific user at the edge of the network as well as for services inside the network. However, the existing work in NFN assumes that the functions are already available on the compute servers and clients always have the raw data to be processed, which is a very limited scenario. A more general scenario of the real edge network is that the raw data, functions and compute resource may reside in different places.

A framework described in [14] provides means to characterize the computation tasks, decide how to distribute, and provide scheduling and caching to meet performance requirements. However, it still assumes that the computation servers already have the requested functions and the network topology info and resource information are a prior knowledge to all nodes in the network.

III. In-Network Compute Orchestration

In this section, we first present basic operation principles of the proposed strategy. Then, we proceed specifying the handling of the Interest packet at network nodes.

A. Operational Principles

We consider an NDN network with the following entities: (i) NDN nodes containing specific data required for the

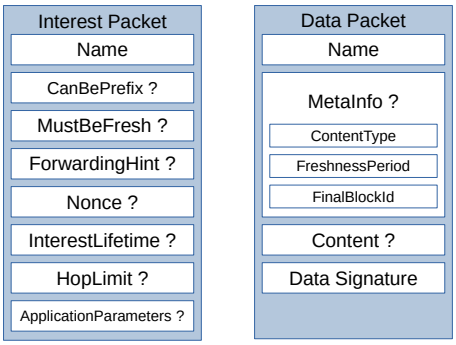
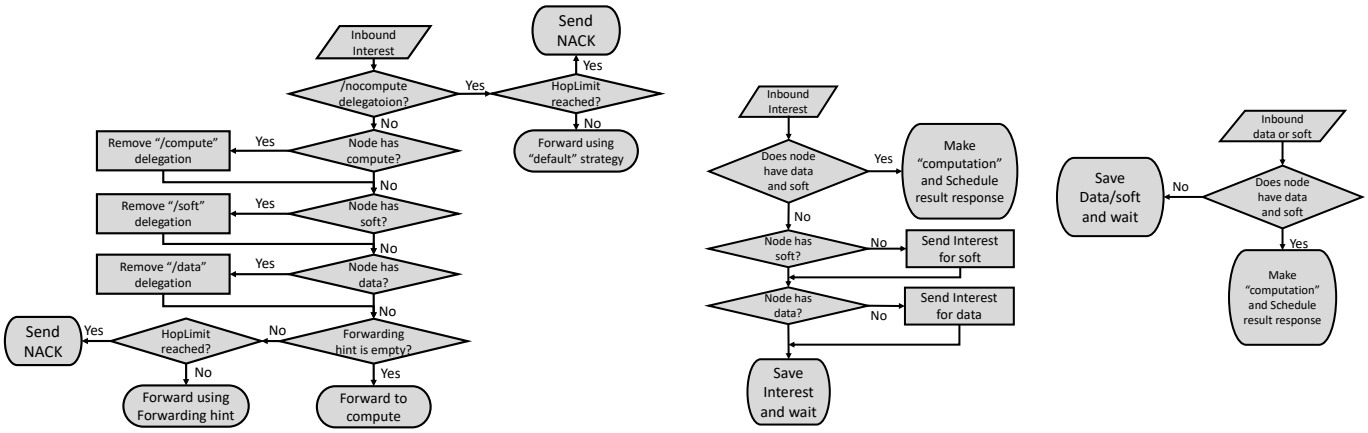


Fig. 2. NDN packet structure.



(a) Intermediate node

(b) Destination node

Fig. 3. Handling of the Interest message at intermediate nodes and destination.

computational task – data producer, NDN nodes containing software required for the computational task – software producer, compute nodes with enough resources for performing a computational task – compute node. By utilizing these entities, a user is assumed to generate a request for processing selected data with selected software on one of the compute nodes available in the network. The only information that needs to be available to the user is the data name and compute function to be executed. Having this knowledge a user could create NDN Interest packet for dynamic compute execution as described below. The benefit of using NDN is that the network layer understands the names of the software, data and compute resources and is able to route the requests accordingly.

The proposed mechanism, called "network makes decision", allows to dynamically combine software, data and select the server to execute computations within the network with limited involvement of the user requesting the service. By utilizing NDN terminology, the node requesting the service is called the consumer, while the nodes providing data, software, computational resources are called *data/soft/compute producers*.

The basic principle of the proposed approach lies in utilization of the *ForwardingHint* field which is an optional field in the NDN Interest packet that, if present, must be handled by the nodes, as shown in Fig. 2. Specifically, we assume that the requested service for the *GenericNameComponent* (GNC) is

$$\begin{aligned} &\textit{result/}\\ &\textit{soft-full-name/}\\ &\textit{data-1-full-name/}\\ &\textit{data-2-full-name/}\\ &\textit{data-N-full-name} \end{aligned} \tag{1}$$

resulting in the overall Interest name

$$\textit{ndn://result/soft\_func\_f1/data\_1/data\_2}.$$

To enable in-network search functionality we utilize the *ForwardingHint* field in the Interest packet that now contains the following field

$$\begin{aligned} &\textit{ResultName}\\ &\textit{?ForwardingHint = /data1/data2/soft/compute}\\ &\textit{?Nonce = Random int}\\ &\textit{?HopLimit = N}. \end{aligned} \tag{2}$$

An example of the trace of the packet traversing the network is shown in Fig. 1(a) while the associated signaling diagram is illustrated in Fig. 1(b). Here, the consumer-generated request for service *result/soft_func_f1/data_1* using the *ForwardingHint* having the following "*?ForwardingHint = [/data, /soft, /compute]*", where "_" acts as a delimiter, which temporary replaces "/". This approach temporarily turns full names of data and software into NDN Name Components enabling intermediate nodes to understand which parts of the full name represent software and data. Note that other types of delimiters can be utilized if the "_" symbol is not acceptable for a particular naming scheme.

As the packet traverses the network, it first encounters data, software, or compute producers. A node that can satisfy one of these sub-names removes the respective Delegation from the *ForwardingHint*. For example, if the data producer is encountered first, the content of the *ForwardingHint* in the Interest packet is modified to *?ForwardingHint = /soft, /compute*, and the packet is forwarded further. When further software producer is encountered, the content of the *ForwardingHint* in the Interest packet is modified to, *?ForwardingHint = [/compute]* and the packet is forwarded next.

If data (software) delegation is missing from the *ForwardingHint*, it indicates that data (software) is available through the Face Interest came from, and FIB should be updated at this and each next node. When the compute node is reached, result-search strategy removes /compute delegation from the *ForwardingHint* and updates FIB. After it the compute node requests software and data by generating individual Interest packets for them. If data and/or software was encountered before reaching compute node, *ForwardingHint* will be missing data and/or software delegation and default routing

should lead towards the Face the Interest came from. In case when data and/or software Delegation is still present in the *ForwardingHint* compute node can decide to send interest(s) anyway or forward the result interest toward data or software. Upon reception of the data and software, the computation is performed and finally, the result is sent back to the consumer satisfying the original result interest.

B. Handling of the Interest at Network Nodes

Handling of the service Interest packet at an intermediate NDN node and at the destination node is shown in Fig. 3. The "network makes decision" is implemented via the "result-search" NFD forwarding strategy that must be installed at the network nodes supporting the service. If the strategy is not installed, a standard forwarding strategy is utilized to forward Interest without changes. If the strategy is not installed, and the node cannot act as *data/software/compute* producer (or does not have these capabilities), the Interest packet is unchanged and forwarded using the default NDN forwarding strategy. Otherwise, the actions are as shown in Fig. 3(a). Other situations, such as reaching the hop limit, are handled in a standard way.

Handling of the inbound "network makes decision" Interest packet at the destination node is shown in Fig. 3(b). Notably, the order in which */data, /software*, and */compute* producers' names are specified in ForwardingHint acts as a suggested order of search of the needed components but does not mandate it. The order can be specified by the user application and may depend on the assumptions of components' reachability, scarcity, size, etc. Nodes utilizing result-search strategy as well as the default forwarding strategy will prefer faces that satisfy routes toward earlier names in the *ForwardingHint*.

Note that as compared to the system, where an edge node has already the service installed, the proposed approach adds the overheads associated with finding the service executables (application). These overheads are standard overheads for an NDN system, However, we note that the knowledge of compute nodes already having an application is generally not available at the consumer or has to be provided by an additional external entity making service provisioning complicated.

We also note that whenever the node who has cached the data removes them for the cache, all the other nodes on the path should update their FIB by removing those relevant faces. However, those nodes would not get notified of the removed content and hence cannot update their FIB, which results in potential miss-forwarding. Thus, the route has to be discovered again by utilizing the flooding mechanism of the NDN system. This shortcoming cannot be addressed without modifying the NDN operation and making the proposed solution incompatible with NDN.

IV. Extensions

The proposed approach allows for several enhancements making it suitable for different use-cases and applications. In this section, we will consider the use of already pre-computed results, function chaining executions, and streaming data.

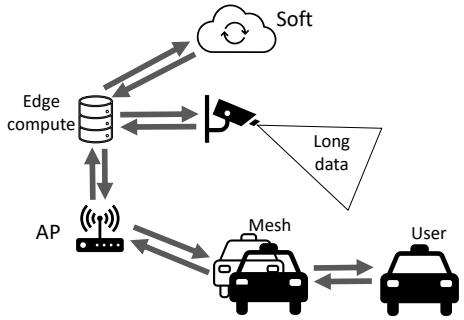


Fig. 4. Use case for streaming data application.

A. Partial data/compute results

The proposed approach can utilize data and or compute results that have already been computed in the network and are available in NDN nodes caches via standard caching functionality of NDN systems.

Since the individual components (data and software) have specific names as well as the full result name, the corresponding Data packets will be cached in the network nodes and can be reused if their freshness is satisfactory. The result-search forwarding strategy upon receiving result interest should not only check FIB for the presence of local producers but also check the cache for data (software.) If the computed result is available at one of the nodes, interest passing through, it will be satisfied automatically by NDN default operation.

B. Function Chaining

The proposed approach can be extended to support function chaining. Here, in the GNC that may look as in (1), where *full-name* can be another result name, for example, resulting in the overall Interest name

$$\textit{ndn://result/soft\_func\_f1/result\_soft}, \tag{3}$$

representing the compute function $f_1(f_2(x_1), x_2)$ in traditional form, where the function f_1 is applied to $x-2$ and the result of the function $f_2(x_1)$ that takes x_1 as an input.

In this case, to enable chaining of compute functions, at compute node, for processing, we need to replace "_" with /, "_" "_" with "_", and "_" "_" "_" with "_" "_", etc, where the number of "_" symbols specifies the nesting level. In the example above, new Interest packets are generated by the compute node in the following form

$$\begin{aligned}
&\textit{Soft interest:soft/func/f1}\\
&\textit{Data interest 1: result/soft_func_f2/data_x_1}\\
&\textit{Soft interest: soft/func/f2}\\
&\textit{Data interest: data/x/1}\\
&\textit{Data interest 2: data/x/2}.
\end{aligned} \tag{4}$$

Handling of the *result/soft_func_f2/data_x_1* Interest packet is similar to the standard "network makes decision" procedure described above. Here, the compute node for the initial Interest acts as a consumer of the result.

C. Streaming Data

The proposed approach can also be used for non-atomic services, for example, for streaming data. Here, the use-case might be related to, e.g., utilizing a video camera to detect suspicious activities and events. Here, the service might be composed as follows, (i) after the first interest server reserves compute for x seconds, (ii) compute sends interest for x seconds of the video stream, (iii) after computing over y, $y < x$, seconds of the video, the server sends the result to the user, (iv) every y new second of video needs a new interest, which updates the reservation. Here, the GNC might look as

$$result/soft\_N/stream\_t{=}x, \tag{5}$$

where x is the absolute of relative timestamp expressed in seconds, N is the software ID.

Then the standard "network makes decision" mechanisms discussed above apply. The proposed extensions for partial data/compute as well as function chaining also apply.

V. Performance Evaluation

In this section, we evaluate performance of the "network makes decision" strategy proposed in Section III and compare it to the user-centric and delegated-node strategies developed in [11]. We also compare the proposed approach with "source makes decision" and "delegated node makes decision" approaches from [11]. The main metric of interest is the fraction of satisfied Interests.

The considered topology represents the standard mobile edge scenario, specifically, a highway/street with Wi-Fi access points (APs) installed along the street. Vehicles, whose passengers request compute service, move along the street at a certain speed. Compute servers are available at both infrastructure nodes and vehicles themselves. The simulated scenario is depicted in Figure 5, where lower part represents wireless mobile part of the network, and all the installed application (i.e. roles) are represented by symbols next to each node. The default "best route" strategy is considered. The simulations have been performed using ndnSIM simulator [15] with all the standard options enabled such as caching. The default system parameters are provided in Table I.

TABLE I
Default system parameters.

Parameter	Values
Number of APs	6
AP spacing	100 m
Number of routers	2
Number of servers per router	2 (4 in total)
Compute nodes per server	1
Number of consumer nodes	100
Number of mesh nodes	50
Spacing between consumer nodes (Mobile)	10 m
Speed of consumer nodes	15 kmph
Number of channels	1
Propagation model	FSPL
Interest frequency 1/s	[1-10]
Decision time	10 ms
Number of computes in mesh/edge	All
Number of cores in mesh/edge servers	1
Edge compute time	10-50 ms
Interest packet Lifetime	2 s
Data/Software size	1MB (667 packets)
Number of data/soft1/soft2 contents	10/10/10
Number of services	100
Cache	50MB
Runs in total	40
Single simulation run time	20 s
Wi-Fi MAC type	AdHoc
Wi-Fi version	802.11ax 5GHz

We start with the basic operation of the proposed approach in Fig. 6 illustrating the satisfied Interest fraction of the proposed scheme and those schemes considered [11]. Note that in addition to the standard proposed strategy we also illustrate the effect of using one optional feature – flooding in wireless (mesh) part of network. As one may observe, the performance of the proposed scheme (Result) is significantly better as compared to "source make decision" and "delegated node makes decision". The rationale is that there is no additional signalling at the wireless edge, and the service is constructed. Implementing flooding feature, however, makes the performance worse, however in some particular cases (e.g., high mobility) flooding may help to satisfy interest that cannot be routed otherwise, but overall utilization of flooding should be limited only to cases where flooding is only "last hope", not to overload the wireless network. Finally, observing the overall performance, one may notice that is degrades drastically with

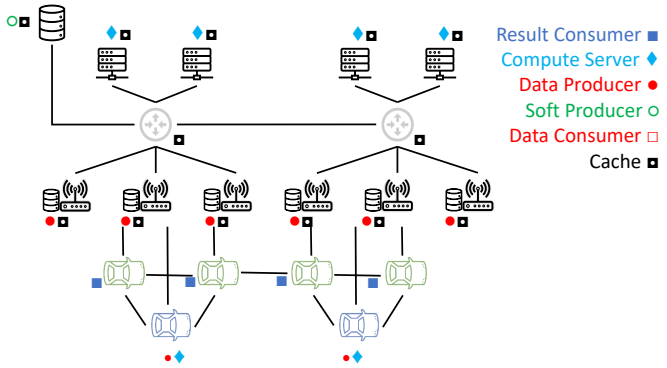


Fig. 5. Illustration of the evaluation setup.

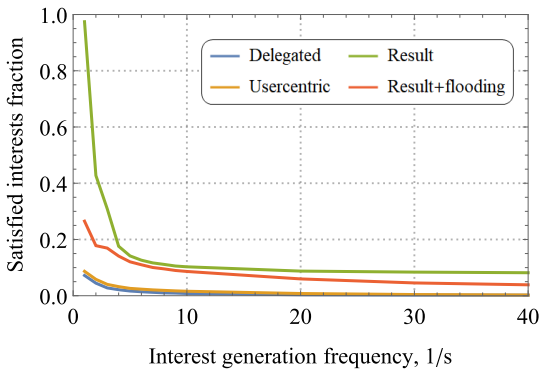


Fig. 6. Satisfied Interests fraction for basic operation.

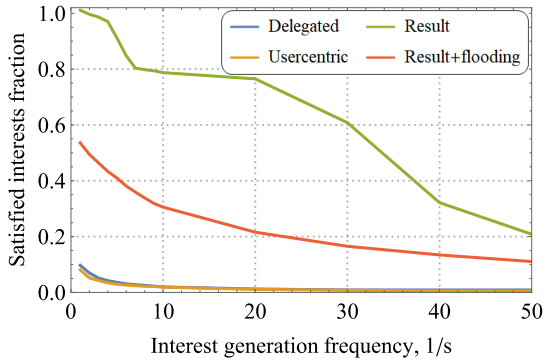
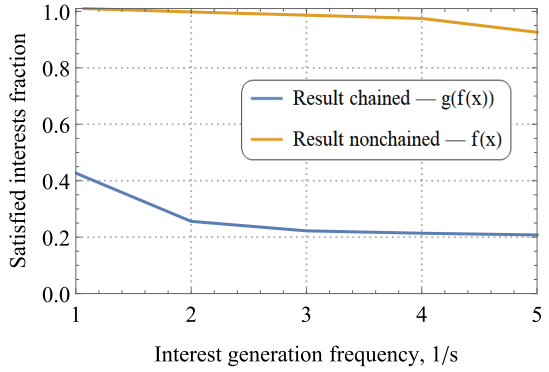
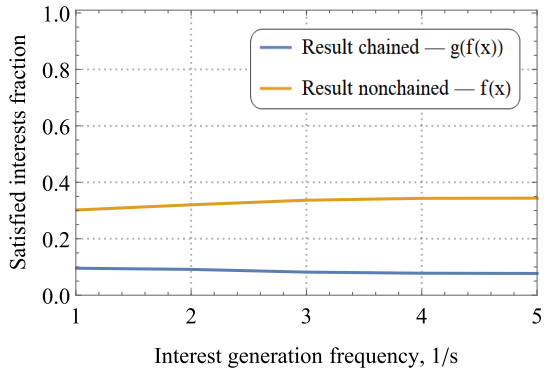


Fig. 7. Performance when partial data/results caching is enabled.



(a) Static scenario

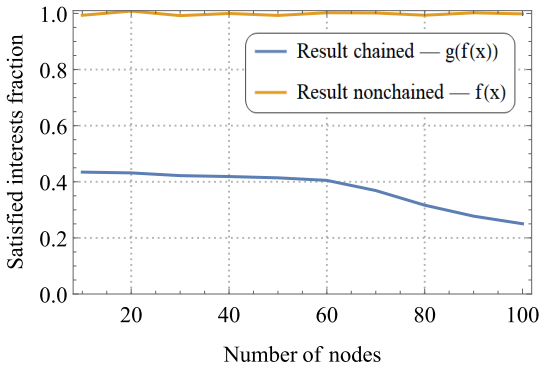


(b) Mobile scenario

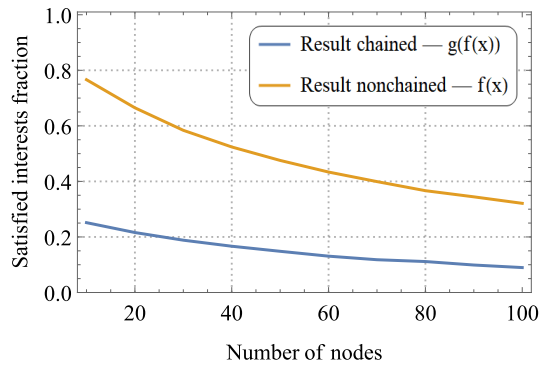
Fig. 8. Performance of chained and conventional compute.

the intensity of interest. This is explained by both inherently unreliable wireless communications and lack of computing resources to serve arriving service requests.

The capability of intermediate node to cache the data they forward is a critical advantage of NDN systems. In context of dynamic compute application, the previously cached data, software and even results that are fully or partially available at intermediate nodes may greatly enhance the proposed of the proposed approach. To showcase the gains of caching Fig. 7 shows the satisfied Interest fraction with enabled caching that allows data and results to be dynamically cached. As one may observe, The performance of the proposed scheme



(a) Static scenario



(b) Mobile scenario

Fig. 9. Performance of streaming compute.

dramatically increases allowing to handle 20 Interest per second arrival rate from more than 100 consumers. The reason is that the bottleneck related to the limited amount of computing resources is efficiently addressed as the results are available is caches in the network. At the same time, performance of the source and delegated node approaches is still low. These observations highlight the importance of outsourcing the compute orchestration to the network itself.

So far we considered static scenario corresponding to traffic jam on the street. Operating over Wi-Fi AP while on the move brings additional packet losses associated with handovers between APs and may degrade the performance. These expectations are confirmed in Fig. 8 that presents comparison between chained and conventional compute according to the proposed approach in Sections III and IV for static and mobile scenarios, where vehicles move at the speed 15 km/h. Here, the chained approach is in the form $g(f(x))$ while the standard operation executes it sequentially by first computing $y = f(x)$ and then estimating $g(y)$.

Discussing the impact of mobility of nodes, we see that for both considered implementations the performance drastically degrades when nodes are mobile. Here, of special interest is the observation that the non-chained approach performs better as compared to the chained one on both static and mobile conditions. The rationale is the complexity of the chained

service resulting in inherent "fragility", where losses at any stage may lead to the failure for the whole service delivery, i.e., N consecutive requests may work better than a chain of N functions in terms of satisfied Interest fraction. In general, in spite its attractiveness, the use of chained in-network services requires careful parameters' optimization.

Finally, we consider the performance of streaming compute illustrated in Fig. 9, where both chained $g(f(x))$ and non-chained, $y = f(x)$, $z = g(y)$, implementations are presented. Here, by following the Fig. 4, we assume that the consumer signs up with a single Interest for a camera stream segmented at a lower level. After the first Interest, producer reserves compute for 3 seconds and then, compute node sends interest for 3 seconds of the stream. After computing over 1 second of the video, the server sends the result to the user. Every second producer needs a new Interest, to update the reservation.

By analyzing the presented results, we observe that the performance of the proposed approach for streaming data is virtually similar to conventional ones. This should not come as a surprise, as the only modifications needed is to dynamically update the reservation. Also, similarly to the results demonstrated in Fig. 8 performance of chained implementation is significantly worse as compared to consecutive requests.

VI. Conclusions

Motivated by the increasing amount of computing resources on the edge and inherently unreliable nature of Wi-Fi access, in this paper we proposed in-network dynamic compute orchestration mechanism for NDN edge/fog systems that requires limited involvement from the consumer in service provisioning and minimizes the usage of wireless infrastructure. We also specified several extensions for the baseline operation including reuse of partially pre-computed results, function chaining and enhancements for streaming computed service.

Our numerical results demonstrate that outsourcing the compute service compositions to the network allows to address the problem of unreliable wireless access while NDN caching capabilities allows to handle extremely high load conditions. Further, the comparison with alternative approaches showing that the proposed method allows to improve the Interest satisfaction ratio by 5-10 times depending on the deployment. We also observed that chained-based implementation is inherently fragile due to complex structure of the service and requires further fine-tuning of parameters.

We note that we evaluated the performance of the proposed approach by utilizing the satisfied Interest fraction as the main metric of interest. Analyzing the edge computing scenarios and also accounting for composite nature of the proposed approach, latency becomes an important metric. We plan to evaluate it in our future studies.

References

[1] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, 2015, pp. 73–78.

[2] M. Iorga, L. Feldman, R. Barton, M. Martin, N. Goren, and C. Mahmoudi, "Fog computing conceptual model," 2018-03-14 2018.

[3] T. Barnett, S. Jain, U. Andra, and T. Khurana, "Cisco visual networking index (vni) complete forecast update, 2017–2022," *Americas/EMEAR Cisco Knowledge Network (CKN) Presentation*, pp. 1–30, 2018.

[4] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos *et al.*, "Named data networking (ndn) project," *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC*, vol. 157, p. 158, 2010.

[5] H. Khelifi, S. Luo, B. Nour, H. Moungla, Y. Faheem, R. Hussain, and A. Ksentini, "Named data networking in vehicular ad hoc networks: State-of-the-art and challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 320–351, 2019.

[6] A. Tariq, R. A. Rehman, and B.-S. Kim, "Forwarding strategies in ndn-based wireless networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 68–95, 2019.

[7] A. Afanasyev, X. Jiang, Y. Yu, J. Tan, Y. Xia, A. Mankin, and L. Zhang, "Ndns: A dns-like name service for ndn," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2017, pp. 1–9.

[8] J. Lee, A. Mtibaa, and S. Mastorakis, "A case for compute reuse in future edge systems: An empirical study," in *2019 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2019, pp. 1–6.

[9] B. Nour, S. Mastorakis, and A. Mtibaa, "Whispering: Joint service offloading and computation reuse in cloud-edge networks," in *ICC 2021-IEEE International Conference on Communications*. IEEE, 2021, pp. 1–6.

[10] M. W. Al Azad and S. Mastorakis, "Reservoir: Named data for pervasive computation reuse at the network edge," in *2022 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE, 2022, pp. 141–151.

[11] R. Pirmagomedov, S. Srikanteswara, D. Moltchanov, G. Arrobo, Y. Zhang, N. Himayat, and Y. Koucheryavy, "Augmented computing at the edge using named data networking," in *2020 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2020, pp. 1–6.

[12] N. Himayat, S. Srikanteswara, K. Bhuyan, D. Guo, R. Pirmagomedov, G. A. Vidal, Y. Zhang, and D. Moltchanov, "Information centric network dynamic compute orchestration," May 19 2022, uS Patent App. 17/598,115.

[13] M. Sifalakis, B. Kohler, C. Scherb, and C. Tschudin, "An information centric network for computing the distribution of computations," in *Proceedings of the 1st ACM Conference on Information-Centric Networking*, 2014, pp. 137–146.

[14] K. Kamran, E. Yeh, and Q. Ma, "Deco: Joint computation, caching and forwarding in data-centric computing networks," in *Proceedings of the Twentieth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2019, pp. 111–120.

[15] S. Mastorakis, A. Afanasyev, and L. Zhang, "On the evolution of ndnsim: An open-source simulator for ndn experimentation," *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 3, pp. 19–33, 2017.