Distributed Compute Offloading in Local Communications

Sameh Eldessoki, Christian Hofmann, Dimitrios Alanis, Panagiotis Botsinis, Tarik Tabet
Apple

***Abstract*—Next-generation wireless networks 6G are tasked with accommodating diverse applications beyond traditional communications. This paper delves into the use-case of computation offload, a pivotal use case within the broader framework of beyond communications applications (e.g., sensing, localization, etc.). More specifically, it focuses on the local subnetwork and decentralized distributed compute use-cases, where the control of the computation offload procedure is not within a central entity. This is realized on top of the subnetwork's architecture. In this work we propose a general architecture and procedural flow that allows for the realization of local subnetwork and decentralized distributed compute.**

***Index Terms*—6G, 3GPP, architecture, networks beyond communications, subnetworks, computational offloading**

I. Introduction

It is anticipated that the demand for higher data rates, lower latency, and increased connectivity continues to surge, along with a multitude of new use cases, such as the immersive education [1]. In this specific use case, both the pupils and the teacher wear *extended reality* (XR) devices and take part in collaborative tasks. In close vicinity, there are several other nodes, such as smart watches, phones or even laptops that have access to the overlay cellular network. In order to satisfy the stringent communication and computational requirements, subnetworks (SubNWs) are deployed to enable local communication as well as computation offload. The focus here is on enabling computation offload, which involves the transfer of computational tasks from resource-constrained devices to more powerful remote servers or edge computing nodes, thereby alleviating the burden on local devices. One special flavor of computation offload is the de-centralized aspect, where the control of the computation offload procedure is not within a specific pre-defined central entity.

Fig. 3 gives an overview of how multiple SubNWs, comprising devices of different capabilities (i.e., low, and high capability devices) can be connected to the cellular network and can enable computational offloading either locally or to a remote edge [2]. For this to be achieved, new control plane (CP) and user plane (UP) procedures need to be introduced, ensuring that the required latency, power consumption, offloaded data accuracy and privacy are met.

In order to leverage on and further enhance the available compute power of local low capability (LC) UEs within a SubNW, it is foreseen that this would require three stages. As shown in Fig. 1 [3], Stage 1 of SubNW creation would involve messaging exchange that enables the connection establishment of local UE(s) with a Management Node (MgtN)

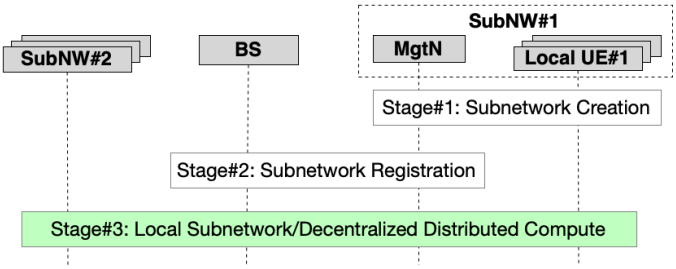

Fig. 1. Proposed three stages procedures for enabling local SubNW and decentralized distributed compute.

and would include multiple sub-stages like synchronization (e.g., Synchronization Signal Block (SSB), Bluetooth pairing) and connection state, such as Idle/Inactive/Connected, Listen-Before-Talk (LBT), etc. Stage 2 would be the SubNW registration phase, which involves messaging exchange between the MgtN and the base station (BS) in order to register the MgtN and its underlying SubNW with the NW. Note that the order of stage 1 and stage 2 can vary; it may be beneficial to perform stage 1 earlier e.g., for providing physical resources from the NW towards the registered SubNW. Finally, stage 3 would involve the messaging exchange between local UE(s) and MgtN of the serving SubNW to enable the offloading of one or more compute task(s) and the reception of the resulting compute result(s). The message exchange could either be done only locally within SubNWs or between different SubNWs in a direct manner without cellular NW involvement (i.e., private SubNWs, private edge nodes) or by involving the underlying cellular NW resources of NodeB (i.e., NW nodes, private SubNWs, private edge nodes or application or cloud servers).

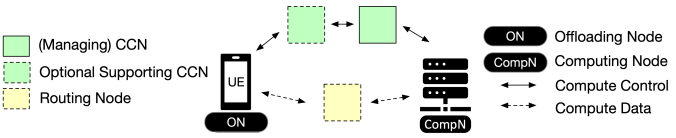

Fig. 2. General functional architecture of distributed compute [3]–[5].

The focus of this paper is on Stage#3: Local Subnetwork/Decentralized Distributed Compute, that would be utilized for some set of special use cases (e.g., immersive classroom) [1]. However, before we go into the new procedures that are needed to enable computation offloading, a new set of general functional architectural components need to be first introduced. Fig. 2 provides an overview of the functional

979-8-3315-2096-0/25/$31.00 ©2025 IEEE

entities consisting of an offloading Node (ON), a computing Node (CompN), compute offload controlling node(s) (CCN) that could be realized as single CCN entity or decomposed into managing and supporting CCN (see in Section III-A) and a routing node (RN) with connections indicating the transfer of computation control and data between the different nodes [3]–[5]. The ON, is a functional entity connected to a wireless network, having a compute task to be offloaded to one or more CompN(s) (e.g., LC or High-Capability (HC) device). The CompN, is a functional entity connected to the wireless network with certain processing capabilities to perform an offloaded compute task and produce compute results (e.g., HC device, core network (CN) function, remote/edge server, etc.). The CCN is another integral functional entity connected to the wireless network that collects all compute capabilities from all available CompNs and makes compute offload decisions based on their capabilities and current load among other parameters. Finally, the RN is the last of the introduced functional entities connected to the wireless network at which the compute task(s) and result(s) get(s) routed between the different ONs and CompNs. In the use case of interest [2], it is expected that the introduced control entity, namely the CCN, would be flexibly deployed within the network and/or SubNWs. This yields that any physical node (i.e., UE acting as a MgtN, UE part of the SubNW, BS, CN, etc.) can take the CCN role.

Our contributions are as follows:

- We introduce the concept of local SubNWs compute offloading and extend it to allow offloading among SubNWs and with the parent 6G network.
- For local SubNW compute offloading, we define all the related message exchange among the MgtN, the ONs and CCNs to enable MgtN-controlled compute offloading.
- As for decentralized compute offloading, we extend the node roles framework by introducing managing and supporting CCNs as well as by defining the message exchange necessary to enable flexible computational offloading within and across SubNWs.

The rest of this paper is organized as follows, in Section II we present an overview on distributed compute paradigms investigated in this paper. In Section III, we focus on local SubNW compute offload and define the processes for its enablement, while in Section IV the case of the decentralized compute offload is investigated, where all the necessary role extensions and message exchanging are defined.

II. Distributed Compute Overview

This section introduces a special type of distributed compute that is tightly coupled with the concept of SubNWs and specifically designed to enable a family of use cases that would require a non-centralized distribution of compute tasks (e.g., immersive classroom). There are two types of compute offload scenarios, each with an expected different procedural flow for compute offload. These two types are “Local SubNW Compute Offload” and “Decentralized Compute Offload”, shown in Fig. 3.

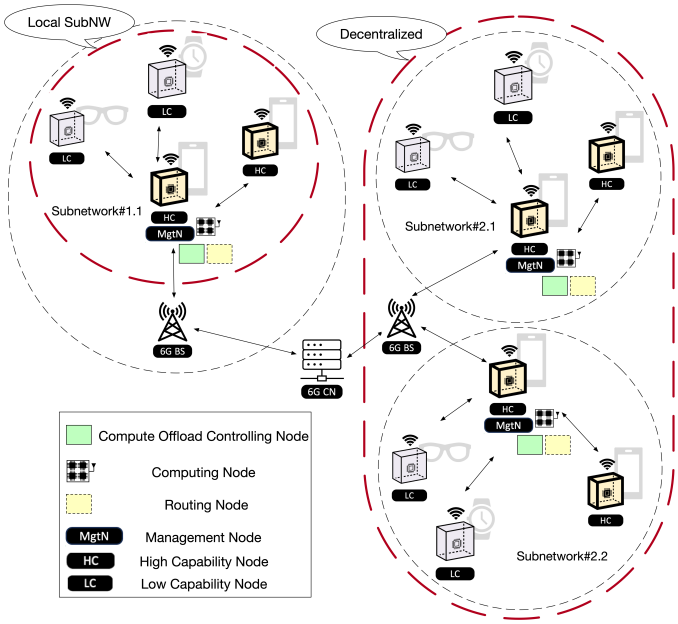

Fig. 3. Local SubNW compute offload and decentralized compute offload.

The first option, the local SubNW compute offload, is where the compute tasks gets offloaded only to the available CompN(s) in the MgtN and/or HC device(s) within the local SubNW without any NW involvement. The negotiation between the different nodes in the local SubNW is done, in order to decide the SubNW CCN among all potential CCNs. Note that this is valid for both MgtN controlled and direct CompN-ON compute offload cases.

The second option, the decentralized compute offload, is where the compute task gets offloaded to any available CompN(s) (i.e., neighbouring SubNW(s), BS(s), CN, cloud server(s), etc.) with or without NW involvement. Negotiations take place among different CCNs in order to decide which node takes the Managing CCN role (i.e., see Section IV-A and Section IV-B later for definition and more details) taking the responsibility in the overall compute offload decision making, and compute task(s) and compute capabilities database(s) maintenance. This can hence be seen as a subsequent stage that comes post the SubNW CCN decision done in the local SubNW compute offload procedure described above.

III. Local Subnetwork Compute Offload

In this Section we focus on the proposed procedural details for the local SubNW compute offload, where the compute task(s) get(s) offloaded only to the available CompN(s) in the MgtN and/or HC device(s) within the local SubNW without any NW involvement. The messaging exchange and hence the distribution control may either be via the MgtN or via a direct ON to CompN exchange. Section III-A goes over a MgtN controlled compute offload procedure. In Section III-B, a proposal is introduced on negotiations between the different ON(s) and CompN(s) in the SubNW to agree on which node would take the CCN role within the SubNW, called a SubNW

CCN. This is then followed by the procedural flow for direct ON to CompN compute offload in Section III-C, that may happen after the SubNW CCN negotiation agreement phase.

A. Management Node Controlled Compute Offload

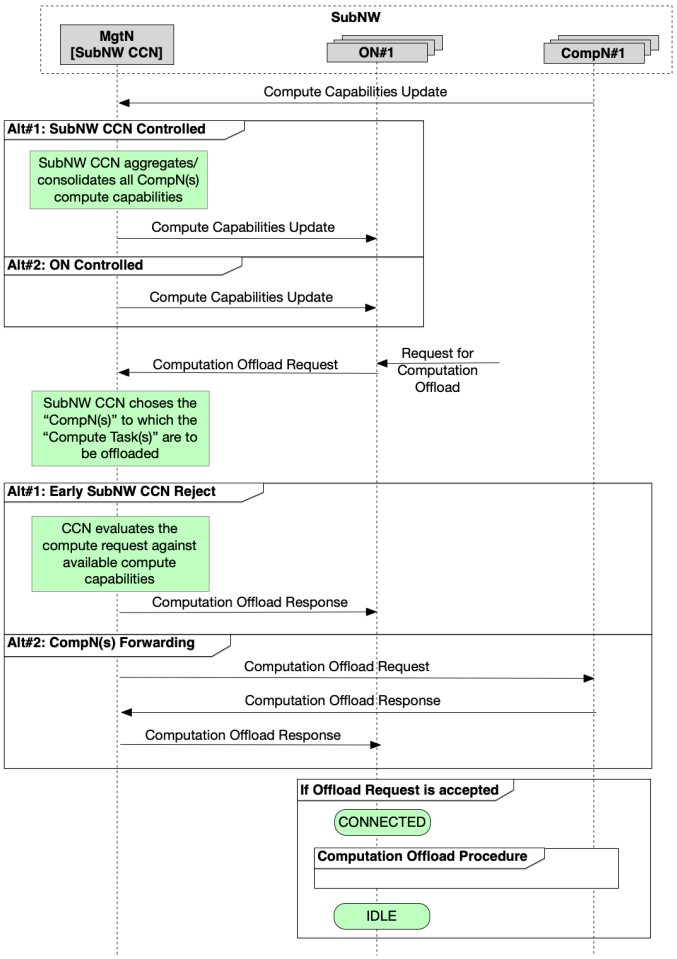

Fig. 4. Management node controlled compute offload message sequence chart.

Fig. 4 shows a high-level Message Sequence Chart (MSC) with messaging exchange between the different SubNW nodes to enable the local SubNW compute offload procedure. In this approach, the MgtN is assumed to be the default CCN entity, without any involvement from any other CCN entities of the NW or that of other SubNWs or other devices within the local SubNW. As mentioned earlier, stage#2 is optional in this computation offload proposal as there is no need for registering to the NW, given that the computation offload is kept within the local SubNW. One option can be that the MgtN is a private HC device that offers support to other local LC devices in its surroundings, forming a SubNW. Another option could be that the MgtN is a NW or 3rd party owned HC device that offers support to other local LC devices in its surroundings, forming a SubNW based on a certain subscription model. In the latter option, for example, the MgtN authentication may take place in stage#1 of SubNW creation without the involvement of other NW or 3rd party entities (e.g., NB, CN, cloud server, etc.). Additionally, in the latter option, the computations distribution is kept within the MgtN control without the involvement of other NW or 3rd party entities (e.g., NB, CN, cloud server).

The procedure starts with all CompN(s) in the SubNW updating the CCN with their compute capabilities, via a "Compute Capabilities Update" message. This message can be sent periodically or event-triggered and can be a broadcast, unicast or multicast message. Alt#1 "SubNW CCN Controlled" in Fig 4 constitutes a centralized approach, where the SubNW CCN aggregates the available compute capabilities and forwards them to all the ONs in the SubNW. A more optimal decision on load distribution is achieved, albeit at the expense of increased MgtN complexity. By contrast, Alt#2 "ON Controlled" in Fig 4 is a decentralized approach, where no aggregation of compute capabilities takes place at SubNW CCN. It is left up to the ONs to choose which of the CompN ID(s) the task(s) would be offloaded to. This alternative provides more flexibility for CompN selection and more privacy, at the expense of a suboptimal load distribution. Once an ON gets a computation task to be offloaded, it would inform the CCN via a "Computation Offload Request" message. This request could optionally include a request for a specific CompN ID to which the ON would like the compute task(s) to be offloaded. The CCN would evaluate the compute request by the ON and decide whether such request can be fulfilled or not. If the request can be fulfilled, the CCN would in turn send a compute request to one or more CompN(s) and wait for their responses. If the CompN(s) accept(s) the compute offload request, by responding with a "Computation Offload Response", the CCN would inform the respective ON and the compute offload procedure would commence. Note that the CCN could respond to the ON with an early rejection of the compute request in case it is already aware that such compute task(s) cannot be fulfilled by the available CompN(s) in the SubNW.

B. Subnetwork Compute Offload Controlling Node Negotiations

Fig. 5 shows an MSC for the SubNW CCN negotiations among the different SubNW entities. The first option is MgtN controlled, where the MgtN decides which of the available CCNs would act as the SubNW CCN and sends indication messages accordingly. Note that the indication message from the MgtN could be extended to provide a list of CCN-IDs that could act as the SubNW CCN (i.e., primary, secondary, etc.), that may be used for fast switch in case of a sudden failure in the primary SubNW CCN (e.g., sudden movement, overheating, etc.). In the second option the MgtN has no central role in the SubNW CCN choice, and it is kept up to the different Nodes to negotiate. The first alternative assumes a broadcast-based approach, where all nodes broadcast their compute capabilities and requests along with their CCN IDs, evaluate the status of the other CCN, and decide on whether to wait to receive or send a "SubNW CCN Indication. The other alternative assumes a request-based approach, where one

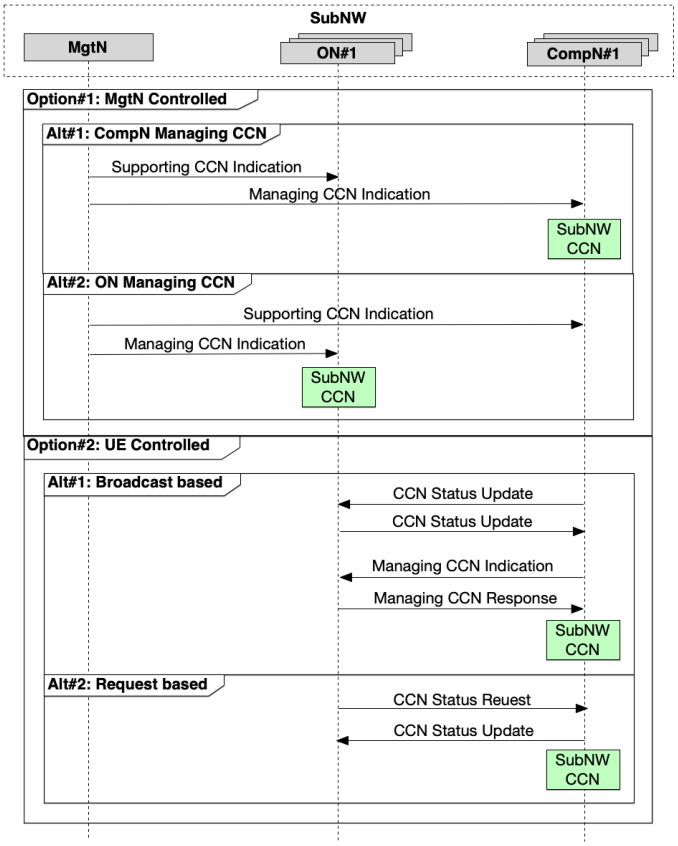

Fig. 5. Subnetwork CCN negotiation message sequence chart.

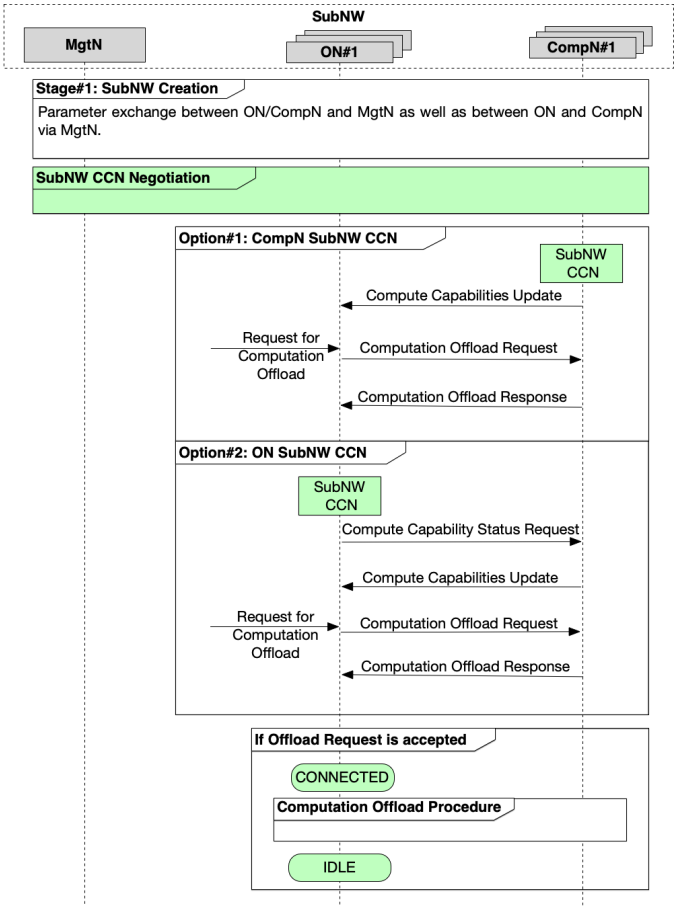

Fig. 6. Direct ON-CompN Compute Offload message sequence chart.

UE would send a "CCN Status Request" indicating compute capability and request along with its CCN ID, while the other UE would respond with a "CCN Status Update" indicating its compute capability/request along with a response on whether it accepts or rejects the role of a SubNW CCN. Finally, note that in case a CompN takes the SubNW CCN role (i.e., as shown in Option#2, Alt#1 and Alt#2), the ON would have to restart the "SubNW CCN Negotiation" procedure if there is a sudden failure in the chosen SubNW CCN (e.g., sudden movement, overheating, etc.).

C. Direct Offloading Node to Computing Node Compute Offload

Fig. 6 shows a high-level MSC for the messaging exchange between ON(s) and CompN(s) to enable a direct ON-CompN computation offload without MgtN involvement. As highlighted in the figure, during the SubNW creation phase, some parameters might need to be exchanged between the ON and CompN via the MgtN, to allow for such direct ON-CompN communication. Those parameters may include transmit and receive resource pools and discovery resources to allow for direct communication in a Sidelink (SL) [6] like manner or assume an LBT to allow for communication between the nodes in a WiFi [7] like manner, or any other proprietary pairing procedure allowing for direct communication between the nodes. Additionally, parameters like CCN IDs, device types and classes, etc. of all devices in the local SubNW are exchanged, to be used by all devices in the SubNW to populate their respective databases and be used later in the direct link SubNW offload procedure. Note that this would then entail a negotiation between the nodes for deciding on who takes control of the compute offload control logic (i.e., SubNW CCN), for which the detailed procedure is introduced in Section III-B.

Following this, there are two possible entities for taking the SubNW CCN role, the CompN or the ON. For the former, CompN updates the ON with its compute capabilities, similarly to before via a "Compute Capabilities Update" message. Once an ON gets a computation task to be offloaded, it would follow the previously introduced steps in Section III-A, by sending a "Computation Offload Request" and waiting for a "Computation Offload Response" from the corresponding CompN(s). For the latter, ON as SubNW CCN, the ON requests information on CompN(s) compute capabilities via a "Compute Capability Status Request". This then proceeds with a "Compute Capabilities Update" message as a response from CompN(s) and a computation offload request and response once an ON gets a computation task to offload, respectively.

IV. Decentralized Compute Offload

In the decentralized distributed compute option, it is assumed that the computation offload is distributed among different NW and/or SubNW nodes. This option entails the involvement of entities such as the NW node and multiple MgtNs of the different SubNWs. This yields that negotiations among the different CCNs of the different nodes are needed, in order to decide the role of each CCN in the computation offload procedure. As a result, an extension of the definitions of the functional entities introduced in Section I and [4], [5] is going to be introduced in the following along with further details on the procedures involved to enable the decentralized distributed compute use-case.

A. Extended Functional Entities Definitions

As shown in Fig. 2 the CCN can be decomposed into two new CCN roles of either managing or supporting CCN. The managing CCN, is a CCN that takes control of the overall computation distribution logic in addition to the management of all available computational resources. On the other hand, the supporting CCN delegates some or all of the compute distribution logic as well as the management of the available computational resources to the managing CCN.

B. Managing and Supporting Compute Offload Controlling Node Negotiations

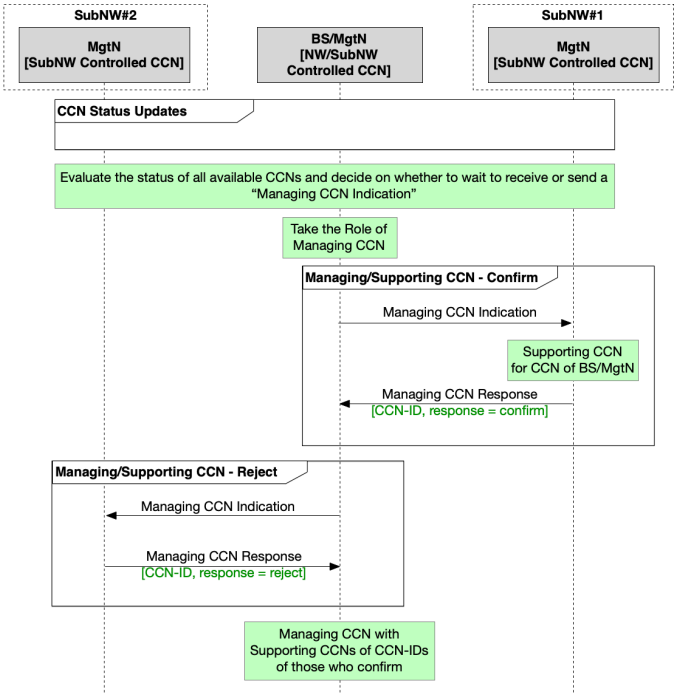

Fig. 7. Managing and supporting CCN negotiations message sequence chart using a periodic or event- triggered status update approach.

This section focuses on the negotiations between the different CCNs for determining which node would act as a managing CCN and which would be a supporting CCN. Fig. 7 highlights a proposal for such negotiations, where all

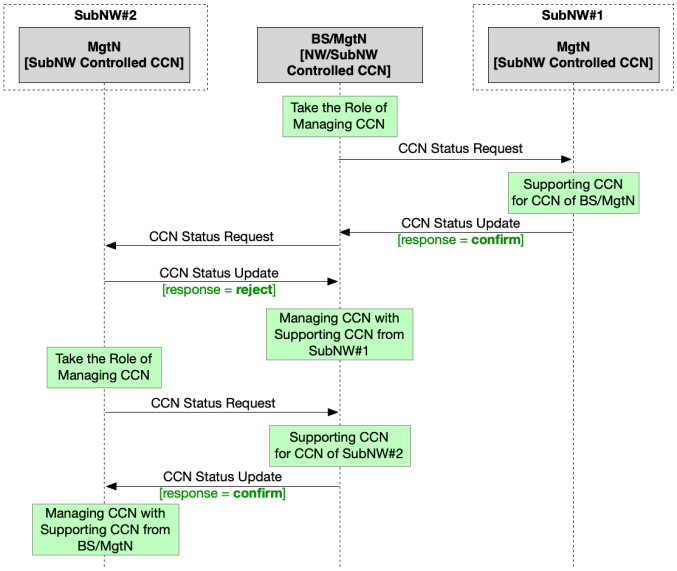

Fig. 8. Managing and supporting CCN negotiations message sequence chart using a request-based status update approach.

available CCNs start by exchanging a "CCN Status Update" message, indicating for example their respective CCN-ID, battery level, compute capacity/requests (i.e., within its local SubNW), compute requests capacity, Rx signal strength, etc. This message could be transmitted either periodically or be event-triggered using broadcast, multicast, or unicast communication (e.g., via system information blocks (SIBs) [8], or dedicated signalling). Based on the NW's or SubNW's own CCN status, as well as on the received status updates of all other available CCNs, each node would evaluate whether to wait to receive or send a "Managing CCN Indication", the latter indicating that it would like to take over as a managing CCN. This message may indicate among other things, the CCN-ID, battery level, compute capacity/requests (i.e., within its local SubNW), compute requests capacity, Rx signal strength, etc.. Each NW and/or SubNW CCN entity that receives the "Managing CCN Indication" message decides whether to confirm or reject the indication via a "Managing CCN Response" message. If a CCN has received a "Managing CCN Indication" from another CCN to which it had also sent an indication, it would have to compare its own metrics with those that it has received and then either "confirm" or "reject" based on which CCN is better. Additionally, if a CCN gets a "Managing CCN Indication" from multiple CCNs, it would compare their metrics, decide the best, "confirm" that one and "reject" all others. As a consequence, the "Managing CCN" entity stores the CCN-IDs of all CCNs that have confirmed its indication, and each "Supporting CCN" entities store the CCN-ID of their "Managing CCN". Note that if the "Managing CCN" disappears and the need for decentralized compute offload is still there, re-negotiations with all available CCNs would take place in order to decide on a new "Managing CCN".

Another approach for realizing the managing-supporting

CCN negotiations procedure is a request-based approach for exchanging CCN status updates. Fig. 8 highlights a proposal for such negotiations, where instead of being periodic or event-triggered, the "CCN Status Update" is sent as a response to an explicit request, namely the "CCN Status Request" message. This message indicates the node's compute capacity and computation requests, received signal strength among other parameters, and also proxies as a request for the targeted CCN to act as its "Managing CCN". In response to the request, the requested node would then respond with a "CCN Status Update" message that would include this node's compute capacity, computation requests, received signal strength among other parameters and would also include a response message either confirming or rejecting the request. If it confirms, then the responding CCN is now acting as the managing CCN of the requesting CCN node, otherwise the requesting CCN would have to request from another CCN via sending another "CCN Status Request" message (i.e., highlighted with the messaging exchange of SubNW#2 in Fig. 8).

C. Supporting Compute Offload Controlling Node Mode

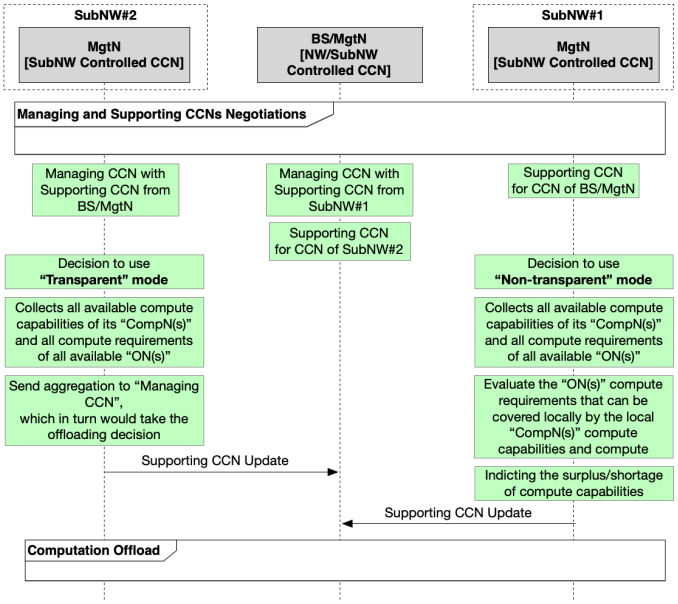

Fig. 9. Supporting CCN mode update procedure message sequence chart.

Now that the negotiations between the different CCNs have been concluded, the mode of the different supporting CCNs shall be decided, where a supporting CCN can either take a "Non-transparent" or a "Transparent" mode. In the non-transparent mode, the "Supporting CCN" evaluates the required compute tasks and available compute resources within the local SubNW and decides which tasks can be locally offloaded. Accordingly, it sends out a request of the remaining compute tasks to the "Managing CCN". On the other hand, in the transparent mode, the "Supporting CCN" simply passes all requested compute tasks and available compute resources within its SubNW to the "Managing CCN", which in turn takes full control on the distribution of the compute offload tasks among all the available compute resources. Fig. 9 shows a message sequence chart highlighting the messaging exchange between the different supporting CCNs and the managing CCN, in which each would evaluate and decide whether it would like to operate in a non-transparent or a transparent supporting CCN mode.

V. Conclusions

In this paper we introduced the general use case of distributed compute (i.e., local and decentralized distributed compute). Additionally, we introduced new functional entities as well as procedures that extend on the state of the art to enable both the local and decentralized distributed compute concepts within and across SubNWs. For the local compute offload we introduced the SubNW CCN Negotiations procedure within the SubNW as well as MgtN controlled and direct ON to CompN offload procedures. For the decentralized compute offload, we introduced new roles of, Managing and Supporting CCNs as well as different supporting CCN modes or transparent and non-transparent modes. Finally, we introduced the negotiations procedure between the different CCNs across the different SubNWs and network entities to choose the managing CCN and supporting CCNs within any setup. As future work, security and privacy aspects should be further investigated to avoid unauthorized access and ensure data privacy. In fact, SubNWs are envisioned to be embedded in the overlay 6G NW, from which the authentication and security procedures could be leveraged.

Acknowledgement

This research is supported by the HORIZON JU-SNS-2022-STREAM-B-01-03 6G-SHINE project (Grant Agreement No.101095738).

References

[1] 6G-SHINE, "D2.1 – Initial definition of scenarios, use cases and service requirements for in-X subnetworks," August 2023.
[2] 6G-SHINE, "D2.2 – Refined definition of scenarios, use cases and service requirements for in-X subnetworks," May 2024.
[3] HEXA-X-II, "Deliverable D3.3 Initial analysis of architectural enablers and framework," April 2024.
[4] 6G-SHINE, "D4.2 - Preliminary results on the management of traffic, computational and spectrum resources among subnetworks in the same entity, and between subnetworks and 6G network," June 2024.
[5] V. Tsekenis and *et al*, "Towards Beyond Communication 6G Networks: Status and Challenges," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, September 2024. Accepted.
[6] 3GPP TS 23.287, "5G; Architecture enhancements for 5G System (5GS) to support Vehicle-to-Everything (V2X) services," July 2022.
[7] IEEE 802.11-2020, "IEEE Standard for Information Technology–Telecommunications and Information Exchange between Systems - Local and Metropolitan Area Networks–Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications - Corrigendum 1 – Correct IEEE 802.11ay Assignment of Protected Announce Support bit," in *IEEE Std 802.11-2020/Cor 1-2022 (Corrigendum to IEEE Std 802.11-2020 as amended by IEEE Std 802.11ax-2021, IEEE Std 802.11ay-2021, and IEEE Std 802.11ba-2021)*, pp. 1–18, December 2022.
[8] 3GPP TS 38.331, "5G; NR; Radio Resource Control (RRC); Protocol specification," August 2022.