Towards Edge Computing Over Named Data Networking

Abderrahmen Mtibaa*, Reza Tourani*, Satyajayant Misra*, Jeff Burke†, Lixia Zhang†
*New Mexico State University, †University of California at Los Angeles

Abstract—This paper discusses leveraging the Named Data Networking (NDN) architecture and Named Function Networking (NFN) to facilitate in-network edge computing. In the NDN context, we consider a the Augmented Reality (AR) use-case—a challenging application—to discuss how NDN functionalities can be leveraged for addressing inherent edge computing challenges, such as efficient resource discovery, compute re-use, mobility management, and security. We present several options to tackle the highlighted challenges and where possible provide solutions.

I. INTRODUCTION

In the IP-based architecture, edge computing applications rely on centralized entities (e.g., proxies and/or SDN controllers) or direct communication with all edge computing devices to make informed task-offloading decisions. These solutions need additional network infrastructure and also introduce overheads—requests and compute resources usage information are routed to proxies or controllers. With increasing application volume and realtime expectations the links connected to these entities become bottlenecks. In these exchanges, the network simply operates as a forwards with the proxy/controller making all the decisions.

Named Data networking (NDN) with it’s name-centric paradigm provides an alternative to the traditional host-centric IP paradigm. We believe NDN is a perfect candidate for provider-agnostic distributed computation such as fog and edge computing. In NDN, in-network computation can be performed at any node in the network that possesses the required compute capabilities and receives a computation request. This allows seamless in-network computation, without the need for the clients to be cognizant of the static mapping between resources and their corresponding IP addresses.

In this paper, we discuss how to create a generic NDN edge computing architecture for secure, efficient and seamless NDN edge computing that leverages NDN networking and forwarding features to offload computation efficiently and seamlessly. We utilize in-network compute execution in NDN using Named Function Networking (NFN) [1] and Named-Function as a Service (NFaaS) [2]—two designs that use function naming to locate remote compute resources and perform in-network computation over NDN. With any edge computing setup there are designs challenges, such as efficient resource discovery, leveraging compute re-use, mobility management, and security. We consider an augmented reality (AR) application as a use-case to highlight how these challenges can be handled in our NDN edge computing architecture.

In the rest of the paper: Section II discusses related work; Section III presents a typical AR application; Section IV deals with efficient resource discovery in order to make accurate in-network offloading decisions; Section V with compute re-use for optimal resource utilization; Section VI with seamless mobility support; and Section VII with secure and private computation for a distributed and trusted execution environment. We highlight these design challenges and discuss potential solutions.

II. BACKGROUND & RELATED WORK

a) Mobile Edge Computing: Various solutions for computation offloading to more powerful surrogate machines [3], known as cyber-foraging [4], have been proposed. The significant impact of large RTT’s on application performance, user’s quality of experience (QoE), and mobile device energy consumption in offloading to distant clouds, has motivated research in edge computing [5], [6].

Edge computing research has so far been focused on task scheduling [7], energy efficient computing [6], and device-to-device edge computing [8]. Most of the existing solutions have focused on applications and services to optimize response time and energy consumption. These solutions do not address networking problems, including how to map application names to IP addresses, or how to leverage the existing network conditions. NDN solves this mapping problem by using application names directly in the network layer forwarding. In this paper, we take this a step further to discuss how NDN enables seamless computing resource discovery, task forwarding, and compute re-use without relying on a centralized entity. Our approach can operate as an optimized standalone solution and also operate with optimizations obtained with the use of proxies and SDN controllers.

Next, we present an NDN primer and discuss NFN to help facilitate the understanding of our design.

b) Named Data Networking (NDN): Different from IP networks that use IP addresses to identify where packets should be delivered to, the fundamental idea of Named Data Networking (NDN) architecture [9], [10] is to retrieve the named pieces of information (named network-layer packets), from any node that can provide it.

Each NDN node maintains three data structures: a content store (CS), a pending interest table (PIT), and a forwarding...
information base (FIB). The FIB (similar to the forwarding table in IP routers) gets populated using a routing algorithm or a self learning scheme [11]. Nodes send an interest packet to receive a corresponding data packet. Any node that receives an interest, performs a CS lookup on the content name in the interest. If the content is not available in the CS, the node searches the name in its PIT. If the PIT lookup is successful, the node adds the incoming interest’s interface to the PIT entry (interest aggregation) and drops the interest. If no PIT match is found, the node creates a new PIT entry and forwards the interest using the FIB to an upstream node (router) in the direction of the data source(s). More details can be found in [9]. Data packets take the interests’ reverse-path to the requester. Upon receipt of data, nodes forward it along the interfaces on which they received the corresponding interest(s).

Named Function Networking (NFN) [1, 12] uses function naming to locate remote compute resources and perform in-network computation over NDN. In NFN, interests are expressions consisting of two components: a routable prefix and an appended expression. This interest will be first routed, as in NDN, to reach the node capable of performing the named computation then the payload data can be extracted and processed to perform further computation or data fetching as needed [13]. NFaaS focuses on function placement and dynamic execution of code using lightweight VMs [2]. This paper focuses on a different design question: how to use NDN naming and aggregation to improve users’ QoE. This paper builds on the general concepts of NDN and NFN in order to enable efficient resource discovery and task aggregation.

III. A TYPICAL EDGE COMPUTING APPLICATION

In this paper, we consider an augmented reality (AR) application as a use-case to highlight our NDN edge computing architecture. We believe that AR/VR applications represent some of the complex applications to be supported at the edge and any solution that handles them well will adapt to the other applications well. Note that, while most of the discussions in this paper are generic and apply to most NDN edge computing applications, few (e.g., compute re-use) are specific to the AR/VR application.

For our AR/VR application, we consider an NDN network of nodes, some have edge computing capabilities. A mobile user with a mobile device (MD) aims at creating an AR view (e.g., annotated view, overlaid images) of a captured scene (also referred to as the Field-of-View (FoV)). If the involved operations are computationally expensive for the MD, the MD schedules the current FoV as a compute task, which will be sent to neighboring edge compute nodes (ENs) for processing.

Fig. 1 depicts a scenario where a mobile device (MD1) is requesting annotation service of its current FoV. MD1 is requesting the service /AR/annotate of the FoV /NMSU/SH/R143/NE/MD1/timestamp/ captured by MD4 at Room 143 of the Science Hall (SH) building at NMSU, while facing north east (NE) at time timestamp. Note that while the network has 4 edge nodes EN_{i\in\{1...4\}} only 3 ENs support the /AR/annotate service, and each of these three have different loads (e.g., CPU workload levels). The goal is for the network to leverage the names and available network level information to route the task to the best EN (among the ENs supporting the requested service).

While conceptually NDN is an edge computing enabler, in the following we will answer questions to bring the concept to its realization. These question include: how to enable seamless task forwarding using NDN/NFN? What constitute the best EN candidate and how to choose it?

IV. RESOURCE DISCOVERY FOR EDGE RESOURCES

Resource discovery (RD) is one of the main challenges in edge computing. In particular, the availability of the resources at an EN may change rapidly due to high compute loads, concurrent clients, and limited resources. These resource availability information must be discovered by intermediate nodes and/or MDs to make efficient task forwarding decisions.

As shown in Fig. 1, ENs must disseminate (1) the services they support (i.e., /AR/Annotate/ or /Word_count/), and (2) their resource availability (e.g., CPU, GPU, memory, storage, and energy available for use). While NDN allows seamless advertisement of name prefixes which will solve (1), one way to disseminate (2) is to make them part of periodic message exchanges (piggybacked on routing messages). In our scenario (Fig. 1), intermediate nodes in the network will forward task request, /AR/Annotate/, to EN_3 as per its lowest CPU utilization.

Neighboring resource discovery (RD) can be performed in three ways: proactive, reactive, and passive.

(i) Proactive RD: In the proactive approach, ENs will proactively advertise their available resources at regular time intervals δt. This approach can guarantee a unified view of the entire network and timely availability of service forwarding entries at any node. However, depending on the period set, this approach can lead to the use of stale information (if δt is large), or a large network overhead (if δt is too small). An interesting option to explore is each EN setting a dynamic advertisement period as a function of the dynamicity of its resources’ availability. In this approach, assessment of the impact (each EN transmits at its own time interval) on application performance and network overhead is important.

(ii) On-Demand RD: In the reactive approach, an MD_1 sends requests for the service /AR/annotate/ to all ENs. On receiving these requests, ENs reply with their load. Intermediate nodes in the reverse path (i.e., Nodes 1, 2, 4, and 5 in Fig. 1)
can update their FIB entries according to the choice of the best EN. This approach generates an overhead proportional to the load in the network, making it suitable only if requests arrive at low frequency. It is also affected by the MD’s mobility and application turnover.

(iii) Reactive RD: A passive approach relies on NDN forwarding plane closed-loop interest-data exchange to send negative acknowledgments (NACK). When an overloaded EN receives a request for a task, it can respond with a NACK to the requesting MD. On receiving this NACK, network routers re-route the task to another candidate EN and can also use this information for routing subsequent interests.

V. EFFICIENT COMPUTATION RESOURCE USAGE

In the context of the image annotation service, users in the same physical vicinity (e.g., attendees in a music concert or visitors attending the Louvre museum) will likely share overlapping FoVs. If these overlapping FoVs will be forwarded to different edge computing nodes, a potential compute reuse and considerable speed-up in task execution time will go unused. Fig. 2 highlights a scenario, where $MD_2$ sends a potentially overlapping FoV to the best edge computing node, $EN_2$ which has the lowest CPU usage. Applying the resource discovery mechanism, described in the previous section, nodes will be oblivious of the existence of a potentially overlapping task, $T_1$, at $EN_2$.

In order to optimize computation resource usage, two solutions may arise: (i) ensuring task coordination at the ENs to redirect tasks to the best EN candidate; or (ii) implementing a new NDN forwarding strategy to forward the task directly to the best EN. Task coordination at the ENs can be relatively easily achieved by connecting the ENs together on the backend (can be part of an extension of the proxy or SDN approaches).

The forwarding strategy based approach can use stored fingerprints (e.g., hash of the task names) of the tasks at the routers for a given freshness period $\Delta t$ to optimize forwarding. This approach can be similar to the fuzzy interest forwarding (FIF) approach and leverage semantic similarities between the names to forward packets [14]. Instead of measuring semantic similarity, another approach can measure the longest prefix match of the task names. For instance in Fig. 2, the similarity score between $T_1$ and $T_2$ will be the highest matching level, i.e., 4, as both task names share the longest prefix /NMSU/SH/R143/NE/ (i.e., levels 1-4 in Fig. 2). This similarity score can be combined (using an objective function) with the cost set at the FIB table to make a more efficient forwarding decision. Therefore, Node 1 and all nodes in the path forward $T_2$ towards $E_2$, which offers a good tradeoff between cost and compute reuse instead of forwarding to $E_3$, which has the lowest CPU usage. Appropriate and autonomous naming scheme is required to efficiently reuse computation. Mobile applications can leverage the MD sensors to get accurate indoor localization and orientation.

This method has the following limitations: (a) if $\Delta t$ is large, similarity score computation can be expensive resulting in a slow-down of the total task completion time, and (b) this method can be exploited by malicious nodes for DoS attacks and/or table poisoning (see details in Section VII).

ENs, if (i) is deployed (and routers if (ii) is used), that receive a potentially overlapping FoVs compare it with other potentially overlapping candidates using image processing features, such as background removal, object detection, image alignment, and stitching [15]. Therefore, ENs extract non-redundant subtasks for annotation (e.g., a moving object within a pre-annotated background scene), and perform stitching and aggregation upon receiving all results from all dependent tasks.

VI. SEAMLESS MOBILITY SUPPORT

In NDN, applications are responsible for re-requesting data upon timeouts. In the context of edge computing, tasks can be compute intensive and execution times can be hard to estimate, resulting in longer timeouts. Also, MD mobility results in connection loss and reconnections that affect response time. The question is how can MD mobility be managed to minimize the task response time?

Fig. 3 illustrates a scenario in which $MD_1$ lost connectivity with Node 1 before receiving the result of its task $T_1$. In NDN, data packets will travel across the reverse path to the destination. Moreover, $MD_1$ can request the data results of its task from the new neighbor Node 3, however Node 3 will not find records for a pending task $T_1$ or which edge computing node is currently executing the task.

We identify two main solutions to these mobility issues: (1) proactive; and (2) reactive. The proactive approach consists of allowing the EN receiving the task, say $T_1$, to append its name to the interest sent to the client. As soon as the client detects a link breakage with its corresponding access point,
it requests the results from the EN executing T1. If the MD disconnects before receiving the first FoV interest, it resends T1 request which may arrive at a different EN from the as the original (the waste of resources here is minimal as the FOV transmission has not happened).

In the reactive solution, when the MD loses connectivity, it re-sends the pending interest. Intermediate nodes closer to the edge computing nodes will find an exact match of the task and forward this new request to the EN currently executing this task or to any cache node in the reverse path that has the result for the task (in case this task is already executed). However, in using this reactive solution, if the NDN network is sparse and ENs are scattered across the network, the task may not reach the EN executing the task. Therefore, we can add a flag to the interest which will be set if the corresponding interest is a re-sent. This flag is checked at intermediate nodes: (i) if no full match of the given task exists then nodes can broadcast it on all interfaces, and (ii) if the exact task interest entry is found, then nodes forward it to the EN executing the task.

This broadcast-based solution can lead to: (i) large overhead in the network, and (ii) malicious use of the broadcast messages. In order to reduce such overhead, we can limit the TTL for these messages to one or two hops assuming that MDs will move to adjacent locations which can be directly connected to the previous access point on the backend.

VII. ACCESS CONTROL & SECURITY

The security challenges include client privacy, Distributed Denial of Service (DDoS) vulnerability, and effective access control enforcement. In this section we discuss two of these challenges: naming client privacy and DDoS vulnerability, which are more relevant in the context of edge computing. As for the access control enforcement, we refer the readers to a comprehensive survey [16] on this matter.

a) Privacy Threats: In NDN, a client requests content by explicitly expressing the content’s name in the interest packet. In our design, the information exposure is not limited to the content name; a request also exposes the location information and the computational operations that should be executed on the content. Potential solutions to anonymize FoV names include: (i) having the MD interact with a network of proxies for anonymity, and (ii) end-to-end encryption between MDs and “trustworthy” ENs [16].

b) DDoS Attack: As discussed earlier, malicious nodes can leverage the use of broadcast messages to perform DoS or DDoS attacks. For EC, we divide the DDoS vulnerabilities into two categories. First vulnerability arises from a set of users requesting various content at high rates with the objective of exhausting the available resources (e.g., CPU or storage resources) at the intermediate routers. The second DDoS attack occurs when a set of users request computationally-demanding tasks from ENs aimed at exhausting the ENs’ resources. While the former attack has been extensively discussed in the NDN community, the latter is specific to EC.

One naive solution to the latter vulnerability requires the routers to consider the ENs’ residual resources in their forwarding decisions. More specifically, a router may perform an implicit rate limiting approach by not forwarding tasks to an overloaded EN, instead, forwarding it to another EN with more available resources. While this approach does not completely solve the DDoS attack on ENs, it limits the snowballing effect and delays impact while corrective actions are made.

VIII. CONCLUDING REMARKS

In this paper, we have discussed how NDN’s features facilitate edge computing. We have identified how efficient resource discovery, compute reuse, mobility management, and secure and trustworthy task offloading can be performed in this context. This work represents one of the first steps towards using named function networking in an AR edge computing scenario. Additional unexplored issues such as scalability, robustness/resilience, resource poverty, energy efficiency, and load balancing to be considered in our future work.

REFERENCES