Named Functions at the Edge

George Xylomenos*, George Pavlou†, Ioannis Psaras†, Ioannis Karakonstantis*
*Department of Informatics, Athens University of Economics and Business, Greece
†Department of Electrical and Electronic Engineering, University College London, UK

***Abstract*—As end-user and edge-network devices are becoming ever more powerful, they are producing ever increasing amounts of data. Pulling all this data into the cloud for processing is impossible, not only due to its enormous volume, but also due to the stringent latency requirements of many applications. Instead, we argue that end-user and edge-network devices should collectively form edge computing swarms and complement the cloud with their storage and processing resources. This shift from centralized to edge clouds has the potential to open new horizons for application development, supporting new low-latency services and, ultimately, creating new markets for storage and processing resources. To realize this vision, we propose Named Functions at the Edge (NFE), a platform where functions can i) be identified through a routable name, ii) be requested and moved (as data objects) to process data on demand at edge nodes, iii) pull raw or anonymized data from sensors and devices, iv) securely and privately return their results to the invoker and v) compensate each party for use of their data, storage, communication or computing resources via tracking and accountability mechanisms. We use an emergency evacuation application to motivate the need for NFE and demonstrate its potential.**

***Index Terms*—NFE, ICN, edge computing**

I. Introduction

In traditional cloud computing, the intensive data processing required by applications takes place on powerful computers located in data centres which process incoming data according to directives by a remote invoker. This mode of operation assumes that the scarce resource is the computation power which, therefore, needs to be hosted in a powerful data centre, with data traveling to the computation. By implication, this model assumes that it is more efficient (in terms of bandwidth) to move the data to the computation, rather than vice versa; that the delay between the invoker, the data centre where the computation is performed and the data generation points, is small; and that it is always possible to reach the data centre, or that it is not critical if the computation cannot be performed.

These assumptions are largely sufficient for mainstream enterprise-level cloud-based applications. However, we increasingly see applications transitioning from a largely "receive-only" mode to a "produce-heavy" mode (e.g., with User-Generated Content from mobile devices, Facebook Live feeds, etc.) and, further, to a "produce-process-actuate" mode (e.g., with sensing and actuating devices, or camera footage). In addition, more and more processing is related to data generated at a specific place (or close to that place) at the edge of the network, rather than to large amounts of data gathered from different places to the core cloud. As we move towards an IoT-dominated environment where fixed (e.g., traffic cameras) or mobile devices (e.g., wearables for health monitors) produce important and privacy-sensitive data, autonomous vehicles produce enormous amounts of data as they travel and user devices can communicate directly with each other in a *Device-to-Device* (D2D) manner, the centralized cloud computing model will need to be re-examined and, inevitably, extended to natively cover the edge of the network.

At the same time, advances in virtualization and container-based networking [1] have made possible the execution of generic computer code on pretty much any platform, thus obviating the need for specialized execution hardware. The amounts of data involved and the low response times required by emerging applications argue for moving the computation closer to the data. However, existing models like cloudlets [2] or serverless at the edge [3] are not suitable, as they are mostly controlled from the cloud, requiring several round-trips to remotely manage the edge infrastructure. Indeed, for many applications, it will soon be: i) much more expensive, ii) much slower, iii) less energy-efficient and iv) less privacy-preserving to move data to functions, rather than move functions to data.

We instead propose a *Named Functions at the Edge* (NFE) platform, supporting dynamic and autonomic computation at the edge, without the mediation of a centralized cloud, by building a market for data, storage and computing between users and edge infrastructure providers. In NFE users will be able to ask for on-demand execution of named functions over named data, letting the system locate, move and execute the functions. This will be done using edge-network resources in a secure and privacy-preserving manner, while ensuring that each party is compensated for the resources it contributes.

The remainder of this paper is structured as follows. In Section II we motivate our work via an emergency evacuation case study. In Section III we review related work and indicate the areas where NFE will be differentiated. In Section IV we describe the elements of our platform, while in Section V we explain how it can be applied to the emergency evacuation use case. In Section VI we present additional case studies for NFE and discuss the business model that could support our platform. We conclude and discuss future work in Section VII.

II. Motivation: Emergency Evacuation

Many natural disasters, including floods, tsunamis, earthquakes and wildfires, require the quick evacuation of the affected area in very adverse circumstances, as disasters often create problems in the infrastructure needed for the evacuation: roads are blocked, electricity grids fail and telecommunication networks are disconnected. Unfortunately, such failures can have cascading effects: even if telecommunication networks withstand the disaster, electricity disruptions will eventually bring them down, reducing network capacity when it is most

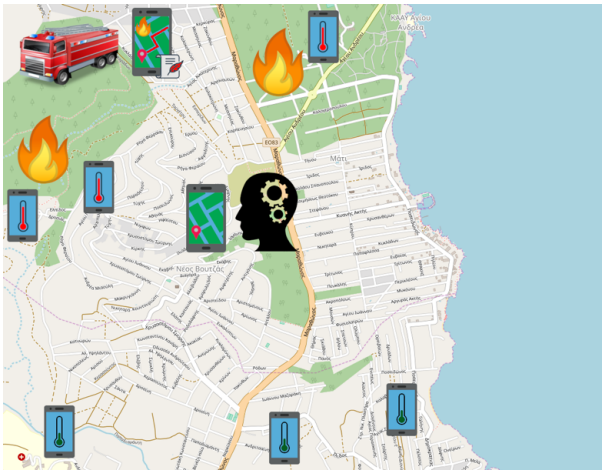

Fig. 1. Emergency evacuation in a wildfire scenario.

needed, while fixing the power distribution infrastructure may be impossible without adequate roads.

Although modern smartphones are equipped with vast amounts of computing and storage resources, GPS chips and detailed maps, they are unlikely to be useful during a disaster. Most applications that could be valuable in the event of a natural disaster depend on network access and cloud-based servers to function. For example, almost all mobile phones have a navigation application whose operation is downgraded (or eliminated) without access to the network; the user may not have downloaded local maps in advance and cannot rely on traffic information which comes from the cloud.

Although in principle applications can be built to help users in the event of natural disasters without requiring access to the network, it is unlikely that a user will have appropriate applications for any possible disaster, let alone the appropriate data for their current location; imagine a user going on holiday having to download applications and maps for every possible disaster in their destination. In addition, an application that works only in disconnected mode cannot use up-to-date environmental information beyond what it can gather from its own sensors. As a result, it cannot take advantage of the potentially valuable information its neighbors have, for example, maps of the area or possible escape routes.

If, however, users could exchange the required functionality (e.g., the evacuation plan for the current disaster), as well as fixed data (e.g., local maps) and dynamic data (e.g., nearby sensor readings for traffic and environmental state) in a D2D manner, their mobile devices could gather the functions and content they are missing, thus saving lives without relying on the cloud or, even, on the cellular infrastructure. In a sense, by extending opportunistic networking, which largely worked for static data exchange only, we envision an opportunistic computing environment for cases where the core network is unreachable, with critical applications depending on dynamic computation. Consider, for example, a user in a wildfire scenario that is spreading to a town next to a forest, as shown in Figure 1: the user could receive the evacuation plan functionality from a fire truck (possibly indirectly, from other users), since local fire fighters are expected to have evacuation plans, while temperature readings from other smartphones can help the user keep away from hot spots.

III. Related Work

Instead of replicating the cloud model at the edge, NFE will allow users to describe a computation (in terms of function and required data), letting the platform locate the code and data and decide where (i.e., at which node) to bring them together. An early approach to this is the Serval architecture [4] for service-centric networking, which resolves service IDs to network addresses in a hop by hop manner, until the data finds a service instance to process them. *Named Function as a Service* (NFaaS) [5] is our earlier approach towards dynamic in-network computations in *Information-Centric Networking* (ICN) [6]. In NFaaS named functions provide results over named content by evaluating expressions, similarly to [7], with the system considering function popularity and latency requirements in order to determine which function to execute at which node. NFE will follow the NFaaS approach of using function names and encapsulating functions in unikernels [8] or similar containers,extending it with function authentication and, more importantly, compensation mechanisms for resource usage.

Locating data in a cloud system requires contacting their points of origin. For a very dynamic system where locality of data is important, it makes far more sense to describe the data (possibly as in ICN) rather than their origins, looking for data first in the vicinity of a function invocation. A tag-based scheme to describe data without enforcing a single naming hierarchy was proposed by [9], where a data item was named by a set of tags, for example, #temp#ucl#engineering#eee. Our own KIOT system [10] greatly simplified that work, providing an efficient routing scheme for locating, gathering and processing data in a hierarchical network. NFE will extend the KIOT scheme for naming and locating data to more complex network environments, adding data authentication and compensation for data provision.

To bring data and functions together, we need to decide where a function should be executed, and how to move data there. This is a complex planning problem, since it needs to consider (1) from the network side: load balancing, latency limitations and bandwidth costs and (2) from the computation side: resource capability and function provisioning (if a function is not available locally, we must download and provision it). We have already explored uncoordinated strategies for function placement that aim to minimize delay and balance load [11]. In addition to considering both data and functions for placement, NFE will also explore the use of market rules for assigning applications to edge-resources and providing compensation to infrastructure providers, following previous work in auction-based resource assignment [12].

A fundamental issue in a large distributed system consisting of edge computation nodes and several infrastructure and application providers, without any centralized authority, is trust and accountability. Data, function and infrastructure providers have to trust each other. In NFE we will consider the use of *Distributed Ledger Technologies* (DLTs), such as scalable

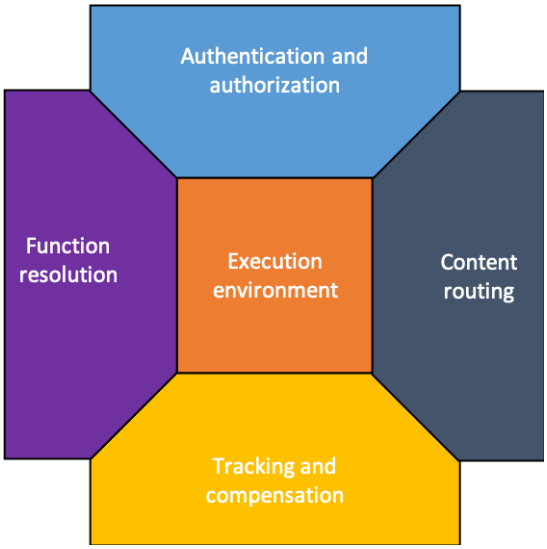

Fig. 2. Overview of NFE building blocks.

blockchain designs, to i) guarantee accountability between trustless peers (e.g., data sources, computation nodes), ii) track and trace data and resource usage and iii) bill for the services provided (data and infrastructure sharing). Essentially, DLTs will provide the infrastructure for building a compensation scheme for resource usage, enabling the creation of a market for data, storage and processing.

Finally, our platform needs to know whether the data to be processed is valid and whether the functions that want to process these data can be trusted. Since we rely on describing functions and data rather than servers, our target is authentication and authorization of content rather than channels, as in today's Internet. *Attribute Based Encryption* (ABE) can preserve end-user privacy [13] and provide access control [14]. As ABE is computationally expensive, we will consider delegation approaches to trusted gateways with higher computation capabilities [15] for edge-nodes with low computational power, such as sensors.

IV. The NFE platform

In NFE both functions and data can move, so that computation can be performed in a quasi-optimal manner, guaranteeing reduced latency at the edge and spreading the computation load across the network. In such a system, many nodes will be able to function not only as data transport intermediaries, but also as trusted function execution platforms and as temporary data repositories. As shown in the Figure 2, NFE requires a generic substrate for (a) describing computations in terms of available functions and data and locating both the function code and the data, (b) determining an appropriate node for execution and moving (if needed) the code and data there, (c) ensuring that the invoker is allowed to use the function to process the required data, (d) executing the function and making the results available to authorized users, and (e) tracking resource usage and compensating resource providers (e.g., data providers, computation and storage nodes).

As both functions and data will need to move, we need a way to refer to both. Name-based approaches have the key advantage of being location-independent and allowing replication and caching of both functions and data, with the closest replica being used. It is also worth pointing out that in highly dynamic environments where mobility does not apply only to end-user devices, but also to network functions, DNS-like approaches fail by default. It is impossible to keep track of where mobile functions are stored, let alone instantiated, in order to redirect requests to that node based on DNS-style resolution. Instead, self-identifiable functions/data should be bundled with other necessary attributes, such as signatures and credentials, so that they may be verified without establishing channels encrypted with *Transport Layer Security* (TLS) with remote servers. We plan to design and implement a name-based network substrate that names and moves both functions and data, building on research advances on name-based resolution [10] and routing and also on our recent work on in-network function execution [5]. This part of NFE will essentially provide a name-based overlay on top of existing network technologies, including both IP networks and link-layer D2D technologies.

The execution environment will be realized as a user-space process on mobile devices and on edge network nodes (access points, base stations, edge routers). We plan to use explicitly named very lightweight code containers, such as micro-VMs in the form of unikernels [8] or JavaScript functions running in container-like environments derived from node.js. Function code will be stored in designated nodes/caches, migrating to other nodes in the edge network based on application demand. In each node capable of executing those functions, a kernel store will be responsible for downloading functions and executing them. Quasi-optimal function placement in edge network nodes can be achieved in a distributed uncoordinated manner according to our previous research [11]. We will also consider different function "service classes", e.g. delay-sensitive or bandwidth-hungry, and a function's service class will be taken into account when deciding on its placement.

Another key aspect of the proposed solution is the operation of a mobile data repository at the edge of the network. Large amounts of data produced by applications will be temporarily stored and processed by functions, with their results feeding additional functions or applications, or temporarily stored, aggregated/anonymized by functions and eventually uploaded to the cloud according to application requirements. Our vision is that storage in edge devices will form an ambient edge micro-cloud with mobile functions operating on the data. This edge micro-cloud will replace the core cloud for applications with tight upper bounds on delay, also allowing disconnected operation. It will also act as an intermediary to the core cloud, preserving user privacy and controlling the upload of data by shaping the upload stream according to network conditions; this is in stark contrast to the current approach where the network is merely a path to the core cloud.

To ensure the viability of our platform, we need to provide incentives to edge-nodes for offering data, storage and computing resources. As a starting point, execution points will participate in a distributed marketplace where application providers can bid for computing resources, similar to [12]. In

turn, application providers will pay infrastructure providers, which we will call *In-Network Service Providers* (INSPs). This way, INSPs are given incentives to invest and maintain the edge computing infrastructure. This can be generalized to include marketplaces for data (e.g., sensor data required for a computation) and connectivity.

To ensure that user privacy is preserved and only authorized functions operate on accessible data, our system will extensively use content encryption, exploiting ABE either directly on nodes with sufficient computational power, or via delegation; for example, IoT devices with limited power will delegate ABE to gateways belonging to the same domain. To ensure that all transactions (e.g., function and data exchange, function execution) are tracked for accountability and charging purposes [16], we will utilize DLTs and smart contracts to allow entities to collaborate with each other in a secure and scalable manner. To ensure accountability, we will only record hashes of transactions to the DLTs instead of the raw transaction data, which would only be revealed to other parties in a case of dispute. We will consider various compensation schemes for resource usage built on top of the DLT.

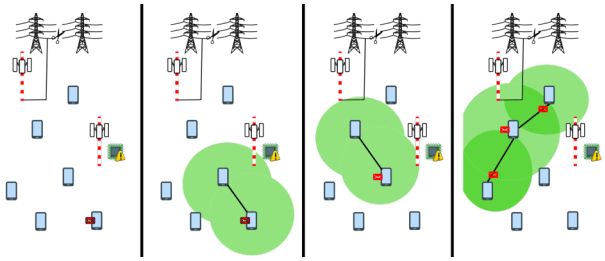

Fig. 3. Store-carry-forward networking with D2D communication.

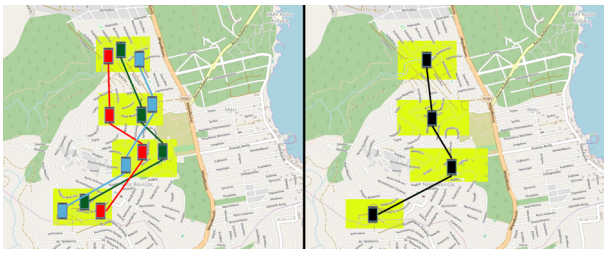

Fig. 4. Generalization of user trajectories.

V. Emergency Evacuation with NFE

The key innovation introduced by NFE for emergency evacuation is that it allows the users to become part of the solution, by using their existing devices (smartphones) to gather any functionality and information needed to plan their evacuation route. The NFE platform, encompassing all the components described in the previous section, will operate as a standalone application with a small footprint in the user's smartphone. The platform will execute different functions depending on the situation, exploiting the already available functionality in the smartphone as well as the capabilities of the NFE platform itself. In this section we explain how NFE can be used in the emergency evacuation case, outlining the data to be exchanged and the functions to be executed.

For communication, the WiFi *Neighbor awareness Networking* (NaN) scheme [17] seems ideal for our goal, which is to quickly identify peers and connect to them for data and function exchange. This technology is only supported on Android 8.0 / 8.1, while devices with standards-compliant hardware are still rare. In addition, WiFi NaN was not designed for multi-party communications or for connectivity to frequently disconnected nodes. Rather than relying excusively on WiFi NaN or some other MAC level protocol for wireless ad-hoc networks, NFE will exploit whatever connectivity is available at the link layer to connect to neighboring users (or to centralized infrastructure, if available), so as to be usable even in devices with outdated hardware and software. Since in the evacuation case the network may be partially disconnected, it will operate in a store-carrry-forward mode using greedy routing algorithms, allowing the movement of both named functions and data, as shown in Figure 3, where information moves from node to node as connectivity becomes available.

In the event of a fire, we would ideally want each user to be able to collect environmental data (ambient temperature, wind intensity and direction) from nearby areas, receive reports from the authorities (fire brigade, police) and run on his or her smartphone a set of functions adapted to the specific case. These functions may be locally available, or distributed by the local authorities (police, fire brigade, civil protection) which have contingency plans specific to that area. For example, the authorities may have designated safe places for people to assemble at or they may have provided preferable evacuation routes. After verifying the validity of such functions (via digital signature checking), they can be propagated to the network in a D2D manner, prioritized over user data.

In order to determine the direction in which the user should move in order to escape the fire, the platform should collect data from participating users. Both the volume and the nature of the data imply the need to aggregate and summarize them, so as to both limit the volume of information to be transmitted to the network and safeguard the anonymity of the participants. In this direction, human trajectory data can be aggregated in each node of the network by a generalization method [18]. This technique ensures both k-anonymity in published data and compression of information. In the generalization technique, initially proposed for collecting human trajectory data, the data of the individual trajectories form clusters with those of the nearest users within a spatial area, as shown in Figure 4, where in the left side we see the individual trajectories and on the right side we see the generalized trajectory. The extent of the area and the number of people who form a cluster satisfy different privacy and precision requirements. Rather than randomly selecting a single trace to represent the cluster as in previous work, we will rank the traces depending on current conditions, so as to select the safest one.

On the other hand, quantitative data collected by sensors of smartphones (e.g. temperature readings) can exploit averaging techniques to provide an overall indication of temperature in an area. When generalizing data which varies over time, for example, temperature readings, the measurements can be weighted depending on their age, so as to make more recent

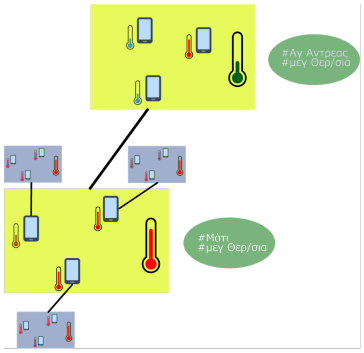

Fig. 5. Aggregation of temperature readings.

observations influence the results more. As shown in Figure 5, each smartphone may aggregate multiple such values from its neighborhood and make them available to others, using appropriate names. To properly aggregate such data, they need to be tagged with various properties, such as time and location where they were generated. The KIOT naming scheme which allows adding arbitrary tags to content, can be especially useful here, as it allows nodes to clearly label their data and quickly identify the data produced by others.

VI. Making NFE Sustainable

In the emergency evacuation scenario discussed above, users are expected to co-operate with each other and the authorities in order to receive the information they need. However, NFE is applicable to many other situations where data locality and latency constraints argue for storing and processing data close to where they are produced. To illustrate the range of possible applications, we provide two additional examples:

- Context-sensitive intelligent transportation: Autonomous vehicles are expected to produce TBs of data from their sensors. Gathering such amounts of data via a wireless infrastructure to centrally provide, say, driving directions that take into account traffic and accidents, is unrealistic, even assuming that users feel comfortable with sharing their data with the cloud or that wireless infrastructure is supported by very high-speed and high-capacity 5G technology. By locally processing and aggregating such data, many applications can operate in a localized manner with a very light footprint and faster response times. For example, navigation systems can employ aggregated data from nearby vehicles and the road infrastructure to optimize a user's route, without the need to know what thousands of cars are doing in an entire city.
- Privacy-preserving edge computing: The practice of sending user data to central cloud infrastructures for processing has recently resulted in multiple privacy violations, and early signs show that users are getting increasingly wary of personal data manipulation. Failing to convince users of the safety of their personal data can make users abstain from such services which would impact the (economic) growth expectations of the industry. Processing personal data, e.g., from wearables, at (or close) to the data source, encrypting them before storage or aggregating them with data from nearby users and sending encrypted bundles to the cloud prevents direct or indirect user-tracking and alleviates fears of data theft.

In such scenarios, any specific application will not offer direct benefits to *all* users, who may be unwilling to provide connectivity, storage and processing resources for the benefit of others. Furthermore, some applications may benefit by the participation of non-user entities that offer their own resources, for example, INSPs or local governments which operate sensor networks, WiFi hotspots or local servers. To attract their participation to the NFE platform, we need to provide incentives for their participation in the NFE ecosystem, which is very different from the cloud-based ecosystem we are familiar with, as well as the societal impact of the NFE model in general.

Currently, the vast majority of the Internet infrastructure is owned, maintained and administered in a closed and centralized manner by very few tech giants. The extraordinary infrastructure that these companies have built formed the ground on which thousands of small companies have innovated, built their business on and created jobs. On the flip side, what is not often mentioned is that due to the centralized nature of their infrastructure, the innovation potential offered by those companies is bound to reach a ceiling, above which growth will decline. The decentralized, function- and computation-centric dynamic edge networks enabled through NFE will provide new capabilities to both users and application developers, thus stimulating innovation.

Paid (and privacy-preserving) vs. free (with targeted advertising) versions of online services are already defining societal classes, with the wealthier being able to afford personal privacy. As we move to an edge-computing dominated Internet, security, privacy and trust issues will be directly linked with quality of life. The NFE model of signed and encrypted functions processing data in secure enclaves is providing the highest levels of security and privacy. Furthermore, transparency of operation is already becoming a selling point for online services and it is bound to become stronger. NFE, by recording compute-related transactions on DLT technology, both allows users to see what happens to their data, and allows them to monetize it directly, in a fine-grained manner.

The largely centralized nature of the Internet makes it impossible to communicate when even a small (but crucial) part of the network is down. With NFE we will showcase through prototype implementation and evaluation/demonstration that dynamic mobile function technology has the potential to bypass the network core and provide life-saving services at the network edge. Through intelligent data and function storage and processing algorithms at the mobile network edge (i.e., on user-devices), we will demonstrate that many applications do not need to rely on centralized infrastructures.

Finally, the explosion of data at the edge of the network is calling for new business models for Internet Service Providers (ISPs). Network upgrade costs to transport data volumes in the area of 50TBs/hour, will overtake the revenue margin of edge/eyeball ISPs who pay higher-tier ISPs to transport customer data towards the core. In addition, as trivial as this may sound, it means that the next generation of vehicles, equipped

with a multitude of sensors, may not participate in the IoT, not due to car-equipment deficiencies, but due to the limits of network infrastructure. The NFE platform will integrate intelligent function and data storage, processing and movement capabilities through its function-centric architectural model. Both storage and processing will be recorded on DLTs to provide transparency of operation. This will help ensure the financial sustainability of a future Internet model, in which data is mainly produced at the edge of the network.

VII. Conclusion and Future Work

We have presented Named Functions at the Edge (NFE), a platform allowing arbitrary code to be executed in a network that can exchange named content and data, using the connectivity, storage and execution resources of devices at the edge, whether these belong to users, government of corporations. We have illustrated the need for such a platform using an emergency evacuation scenario as a use case, although many other scenarios can benefit from such an approach. Beyond presenting the main components of our platform and their use in the case study, we also discussed the business aspects of the NFE approach and the incentives that will ensure the participation of different entities to it.

We have already prototyped and modeled key aspects of NFE in our previous work, including the NFaaS model for computation in the network and the KIOT model for tag-based routing. Our next step is to integrate these components with a DLT-based tracking system and an access-control and authorization scheme, followed by an implementation of the NFE prototype for Linux and Android, so as to allow it to be used with smartphones and smart WiFi access points for larger scale deployments and trials.

Acknowledgments

This research was supported by the RC-AUEB funded "Original Scientific Publications" project under contract ER-3013-01.

References

[1] A. Madhavapeddy, T. Leonard, M. Skjegstad, T. Gazagnaire, D. Sheets, D. Scott, R. Mortier, A. Chaudhry, B. Singh, J. Ludlam, J. Crowcroft, and I. Leslie, "Jitsu: Just-in-time summoning of unikernels," in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2015, pp. 559–573.

[2] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Cloudlets: Bringing the cloud to the mobile user," in *Proceedings of the ACM Workshop on Mobile Cloud Computing and Services (MCS)*, 2012, pp. 29–36.

[3] S. Hendrickson, S. Sturdevant, T. Harter, V. Venkataramani, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Serverless computation with openlambda," in *Proceedings of the USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*, 2016.

[4] E. Nordström, D. Shue, P. Gopalan, R. Kiefer, M. Arye, S. Y. Ko, J. Rexford, and M. J. Freedman, "Serval: An end-host stack for service-centric networking," in *Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI)*, 2012, pp. 7–7.

[5] M. Król and I. Psaras, "NFaaS: Named function as a service," in *Proceedings of the ACM Conference on Information-Centric Networking (ICN)*, 2017, pp. 134–144.

[6] G. Xylomenos, C. N. Ververidis, V. A. Siris, N. Fotiou, C. Tsilopoulos, X. Vasilakos, K. V. Katsaros, and G. C. Polyzos, "A survey of information-centric networking research," *IEEE Communications Surveys Tutorials*, vol. 16, no. 2, pp. 1024–1049, Second 2014.

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

[8] A. Madhavapeddy and D. J. Scott, "Unikernels: Rise of the virtual library operating system," *Queue*, vol. 11, no. 11, pp. 30:30–30:44, Dec. 2013.

[9] M. Papalini, A. Carzaniga, K. Khazaei, and A. L. Wolf, "Scalable routing for tag-based information-centric networking," in *Proceedings of the ACM Conference on Information-Centric Networking (ICN)*, 2014, pp. 17–26.

[10] O. Ascigil, S. Reñé, G. Xylomenos, I. Psaras, and G. Pavlou, "A keyword-based ICN-IoT platform," in *Proceedings of the ACM Conference on Information-Centric Networking (ICN)*, 2017, pp. 22–28.

[11] O. Ascigil, T. K. Phan, A. G. Tasiopoulos, V. Sourlas, I. Psaras, and G. Pavlou, "On uncoordinated service placement in edge-clouds," in *Proceedings of the IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, Dec 2017, pp. 41–48.

[12] A. G. Tasiopoulos, O. Ascigil, I. Psaras, and G. Pavlou, "Edge-MAP: Auction markets for edge resource provisioning," in *Proceedings of the IEEE International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, June 2018, pp. 14–22.

[13] M. Ion, J. Zhang, and E. M. Schooler, "Toward content-centric privacy in ICN: Attribute-based encryption and routing," in *Proceedings of the ACM SIGCOMM Workshop on Information-centric Networking (ICN)*, 2013, pp. 39–40.

[14] B. Li, D. Huang, Z. Wang, and Y. Zhu, "Attribute-based access control for ICN naming scheme," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 2, pp. 194–206, March 2018.

[15] N. Fotiou, A. Machas, G. C. Polyzos, and G. Xylomenos, "Access control as a service for the cloud," *Journal of Internet Services and Applications*, vol. 6, no. 1, p. 11, Jun 2015.

[16] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, May 2016, pp. 839–858.

[17] D. Camps-Mur, E. Garcia-Villegas, E. Lopez-Aguilera, P. Loureiro, P. Lambert, and A. Raissinia, "Enabling always on service discovery: WiFi neighbor awareness networking," *IEEE Wireless Communications*, vol. 22, no. 2, pp. 118–125, April 2015.

[18] M. E. Nergiz, M. Atzori, and Y. Saygin, "Towards trajectory anonymization: A generalization-based approach," in *Proceedings of the ACM International Workshop on Security and Privacy in GIS and LBS (SPRINGL)*, 2008, pp. 52–61.