# Key ingredients in an IoT recipe: Fog Computing, Cloud Computing, and more Fog Computing

M. Yannuzzi*, R. Milito†, R. Serral-Gracià*, D. Montero*, and M. Nemirovsky‡
*Networking and Information Technology Lab (NetIT Lab), UPC, Barcelona, Spain
†Cisco Systems, San José, CA, USA
‡Senior ICREA Researcher at Barcelona Supercomputer Center (BSC), Barcelona, Spain

***Abstract*—This paper examines some of the most promising and challenging scenarios in IoT, and shows why current compute and storage models confined to data centers will not be able to meet the requirements of many of the applications foreseen for those scenarios. Our analysis is particularly centered on three interrelated requirements: 1) mobility; 2) reliable control and actuation; and 3) scalability, especially, in IoT scenarios that span large geographical areas and require real-time decisions based on data analytics. Based on our analysis, we expose the reasons why Fog Computing is the natural platform for IoT, and discuss the unavoidable interplay of the Fog and the Cloud in the coming years. In the process, we review some of the technologies that will require considerable advances in order to support the applications that the IoT market will demand.**

***Index Terms*—IoT, Fog Computing, Cloud Computing, mobility, data analytics, real-time control, actuation, security.**

## I. Introduction

In the next few years, more and more "things" will produce and consume data in ways that we are just starting to envision. Many of these "things" will be part of much larger systems, which will obviously require compute and storage capacity for processing and storing their data. As a matter of fact, deciding the location of such resources poses non-trivial questions to platform designers. In the first place, the majority of IoT endpoints will be very basic, which means that they will not embed the compute and storage resources that they need. In other words, external means will be required in order to perform a considerable part of the data processing tasks. Secondly, the strategy for placing those "external resources" is not self-evident. The demands for compute and storage resources will probably come from tens of billions of fixed and mobile endpoints, which will span vast geographical areas and will be arranged in various different forms, covering a plethora of different use cases and settings. In turn, many of these settings will have stringent requirements, such as very low latency, high throughput during short time periods, prompt decision making based on real-time analytics, and different combinations of these and other requirements as well. As if that were not enough, many IoT ecosystems will be characterized by other quite restrictive constraints, such as low power communications, scarce energy, lossy communications including signal fading problems, short reach radio, etc.

In summary, IoT will demand significant compute and storage resources, but the question of where should those resources be placed remains open. The success and growth of Cloud Computing would probably make an eager respondent say: "In the Cloud, of course". Unfortunately, the requirements and design space of IoT make Cloud Computing unfeasible in numerous scenarios, especially, when the goal is to build a general and multipurpose platform that can serve a wide variety of IoT applications. To illustrate this point let us consider the following:

Are there applications in the IoT space that would require offloading some computation tasks onto other devices, but the mere fact of having the compute and storage resources confined in a centralized data center makes the overall platform unfeasible?

The answer to this question is "Yes", and the main reasons lie on Fig. 1. Observe that the number of potential use cases foreseen under these sets is huge, and particularly covers applications where the offloading of computation and data processing tasks is essential, but their characteristics make them particularly problematic to run in "the Cloud". In this paper we analyze these sets, their requirements, and based on them derive the reasons that unquestionably support the fact that the recipe for building scalable IoT platforms is the following: a) add "Fog Computing" [1], [2]; b) add "Cloud Computing"

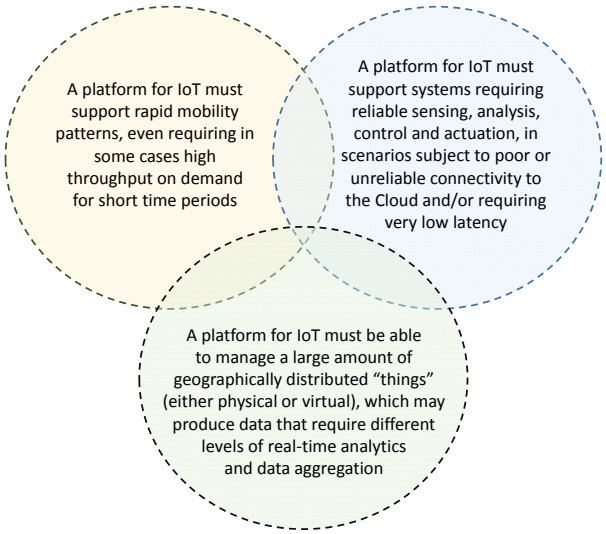



Fig. 1. Some of the main requirements for designing and building an adaptable and scalable IoT platform.

and smartly combine it with the Fog, and c) whenever the platform needs to scale either to cover more "things" or more Point-of-Presence (POPs), just add more Fog nodes.

In a nutshell, Fog Computing proposes a model in which data can be analyzed and processed by applications running in devices within the network rather than in a centralized Cloud. Indeed, by smartly orchestrating and managing compute and storage resources placed at the edge of the network, Fog Computing can deal with the ever-increasing number of connected devices and the emerging demand in IoT. As we shall show, as long as the technological requirements and constraints of the IoT applications are properly fulfilled, it is up to the platform designer to decide whether an endpoint should be served by the Cloud, the Fog, or an adequate combination of the two at any given time during the service lifetime. As the economy of scale dictates, whenever possible Cloud is the right venue. Hence, Fog Computing is not devised as a competitor of the Cloud; quite on the contrary, it is envisioned as the perfect ally for a broad spectrum of use cases and applications for which traditional Cloud Computing is not sufficient. These two technologies are called to interplay and benefit from each other in a synergistic way, since only a smart combination of communications, orchestrations, and assignment of compute and storage resources can tackle the requirements of IoT.

The rest of the paper is organized as follows. Section II examines the challenges posed by mobility in IoT, and particularly discusses the limitations of Mobile Cloud and other enabling technologies, such as LISP and Linux containers. Section III examines the issues related to reliable control and actuation, while section IV analyzes the challenges behind data aggregation, and the design of scalable models enabling real-time analytics and decision making. Section V examines the strengths of Fog Computing under the challenges exposed in previous sections. Finally, section VI concludes the paper.

## II. Mobility in the Coming Age of IoT

The set on the upper left of Fig. 1 covers IoT applications that need compute and storage support on the move, and more importantly, such support could be required under rapid mobility patterns. Mobility, and especially fast mobility, is one of the aspects in which the traditional Cloud paradigm needs an ally to achieve pervasiveness, while offering the reliability required by the applications. In particular, under rapid mobility it makes much more sense that the computing intelligence required "moves" with the "things" that need it—by "moves" we mean either physically or virtually. This is because those moving "things" may traverse large geographical areas in a few minutes, so the interactions with compute and storage resources will be subject to dynamics that the traditional Cloud model is not yet ready to offer. Potential scenarios subject to these requirements could involve applications for sensors moving with a vehicle, for passengers while commuting in a metro, a train on the countryside, an aircraft, etc. Clearly, the deployment of more and more Cloud facilities in the traditional way, and distributing them among large geographical areas will not solve the problem.

### A. Mobile Cloud Computing

One of the initiatives that is trying to tackle part of these requirements is Mobile Cloud Computing (MCC) [3]. This paradigm aims at enabling compute and network services ubiquitously with the help of the Cloud (see Fig. 2). Issues such as the limited amount of resources of many mobile and M2M devices (e.g., battery duration, processing power, memory, etc.) are clear drivers for offloading part of the computation and data processing tasks onto the Cloud. Unfortunately, the MCC paradigm is especially challenging for applications with the requirements shown in the set at the upper left of Fig. 1. On the one hand, MCC suffers from the intrinsic characteristics of mobility, such as frequent variations of network conditions, including signal fading and severe service degradation. Observe that these issues are clearly amplified under rapid mobility patterns. On the other hand, even if the mobile device remains in a fixed location, the communication path to reach the Cloud is often subject to bandwidth limitations and communication latency, and these issues strongly depend on the radio coverage, the interference, and the amount of resources shared with other mobile devices.

An alternative in this regard is to rethink the way in which cells and wireless networks are currently deployed, and bet on Heterogeneous Networks (HetNets) [3], which offer a variety of radio access nodes and technologies under the umbrella of the Long Term Evolution (LTE) network. As shown in Fig. 2, a mix of macro, pico and femto cells, as well as other possible configurations, offer a promising approach to meet the traffic rates and demands of broadband wireless networks. Unfortunately, even the combined MCC and HetNet model cannot deal with rapid mobility patterns. This is because the macro cells are devised to provide coverage to wide areas, and therefore avoid frequent handovers. However, this comes at the cost of the relatively low data rates and signal instability. Pico and femto cells, on the other hand, provide much higher data rates and signal stability, but their coverage is restricted to a

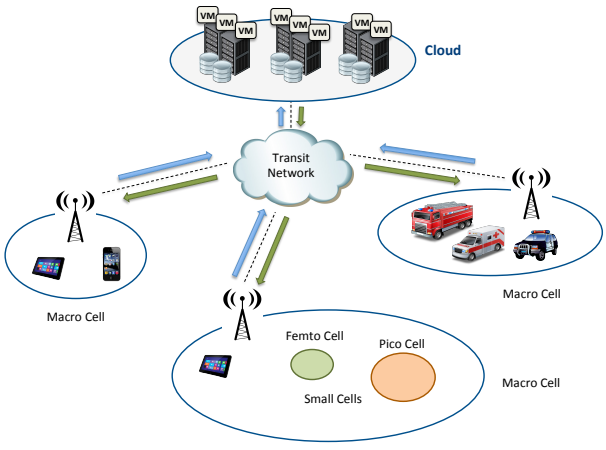



Fig. 2. A Mobile Cloud Computing (MCC) scenario. The arrows toward the Cloud represent computation requests, data uploads, etc., while the ones in the opposite direction represent computation results, commands or actions, data downloads, etc.

few hundred meters, thus they are not conceived for serving applications under fast mobility.

Let us now proceed to analyze other potential technology enablers, and examine some of the existing limitations to truly unfold the mobility of physical and virtual devices under rapid mobility patterns in IoT.

### B. Mobility of Virtual Containers

One of the challenges that fast mobility imposes on IoT platforms is the capability to provide and keep the compute and storage resources close to the "things", and, in the event of a sudden change in the location, the platform must be able to rapidly reconfigure the switching context, and seamlessly orchestrate the allocation of resources and the migration of the state to the new location. To address this challenge, resource virtualization techniques allow the creation of isolated execution contexts that can be allocated for the processing needs of the "things", and migrate them whenever required. Resource virtualization techniques based on containers offer a promising approach in the context of IoT, since they have the potential to enable lightweight migrations, such as just the container state. LinuX Containers (LXC) [4], Docker [5], or CRIU [6] are some representative examples, but further advances are still needed. For instance, at the moment CRIU only supports process migration—we will assess this in Section II-D.

### C. Addressing Considerations

Many of the IoT use cases foreseen thus far require the survival of ongoing sessions even in the event of switching to another L2/L3 network (e.g., due to a handover). Indeed, one of the main challenges is to ensure the same level of security irrespectively of the physical location—observe that this includes the application and maintenance of security policies and rules on the move. To cope with this challenge, the most important technologies currently addressing end point mobility with session continuity are the Locator/Identifier Separation Protocol (LISP) [7], and Multi-Path TCP (MPTCP) [8]. LISP allows an endpoint to switch between networks while keeping its Identifier (IP address) intact—by maintaining the Routing Location (RLOC) of each Identifier in a mapping system, which is updated by its control plane. On the other hand, MPTCP defines TCP sub-flows at the transport layer based on the IP addresses of all the enabled interfaces on a device. Under mobility, whether the device changes its IP or switches radio technologies (e.g., WiFi to 4G), the new IP address is registered and a new sub-flow is opened. This strategy allows for seamless mobility of the device across networks and radio technologies. Clearly, both LISP and MPTCP are solid technologies for enabling mobility in IoT.

### D. Limitations in terms of Handover and Live Migrations

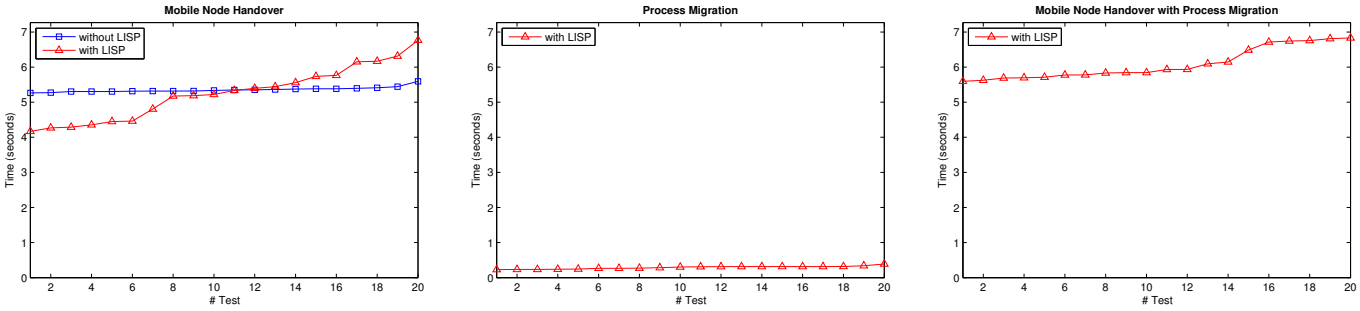
In order to analyze the capabilities and limitations of state-of-the-art technologies for some of the mobility cases foreseen in IoT, we have run a set of experiments considering three different mobility scenarios: i) Mobile Node Handover, ii) Process Migration using CRIU, and iii) Mobile Node handover with Process Migration. The first one is a typical scenario where a device on the move changes Access Points (APs), forcing a handover at the endpoint. The second case is oriented to process migration from one domain to another (e.g., due to scheduled maintenance), and in the last one, we consider the case when both the mobile endpoint and the process move (e.g., the application requires very low latency and the endpoint is moving very fast, thus the offloaded computation process needs to move along with the endpoint).

For each scenario, we consider the case where a connection between the mobile endpoint and the process needs to be maintained, and this applies even during the process migration or the handover process (in our case, seamless handover/process migration using LISP). This case actually represents a typical stateful protocol where both ends of the connection need to keep state. Furthermore, for the first scenario we also tested the case where the connections are stateless and self-contained. This could stand for a typical data harvesting sensor—probably a proxy of a larger sensor network—that temporarily stores data that is pushed to a server periodically without real state between pushes. To implement and test these scenarios, we have set up a testbed composed by two different APs configured in different networks, and a computing node in each site providing service (represented as a process) to a mobile node, i.e., an Android cell-phone in this case using 802.11g.

The purpose of the experiments is to measure the downtime due to a mobile node handover and/or a process migration. For each scenario, twenty independent experiments were made and the results were sorted in ascending order according to downtime obtained. Figure 3(a) shows the results for the first scenario considering both cases, the stateful and stateless connections including the DHCP overhead at the end-point side. The results show that for the stateless connections, where LISP is not required, the connection disruption is around 5.0 seconds—this time encompasses the end-point handover and the application reconnection. On the other hand, for the stateful connection which implies that both end-points implement LISP, the results show that the connection disruption may vary between 4.0 and 7.0 seconds approximately. This variation is due to two main reasons: i) the end point does not drop the TCP connection stack and hence reduces the downtime; and ii) the overhead introduced by the LISP control-plane is variable, and ends up impacting on the overall downtime. Mainly, the LISP control plane has to notify when the location of one of the connection ends has changed to the other connection end.

Figure 3(b) presents the results for the second scenario considering a stateful connection with LISP support at both ends and no DHCP (i.e., the new location identifier of the process is statically configured). Observe that the disruption incurred by a very lightweight process migration is around 300 milliseconds. Finally, Fig. 3(c) reports the results for the third scenario (i.e., the process moves along with the end-point, and both ends of the connection are LISP enabled). As it can be observed, the downtime varies between 5.6 and 7.0 seconds approximately.

Overall, even a shallow analysis of some of the most natural




(a) (b) (c)

Fig. 3. Service disruption due to: (a) Mobile Node Handover; (b) Process Migration with CRIU; (c) Mobile Node Handover with Process Migration.

technologies for enabling mobility in IoT reveal that, considerable advances will be needed in the field in order to support rapid mobility of "things" at the edge.

## III. Reliable Control and Actuation including Real-time Operation in IoT

The set on the upper right of Fig. 1 covers fundamentally two niches in which the traditional Cloud Computing model falls short. The first one includes systems that are placed in locations where the communications with the Cloud are either too expensive or they are simply unreliable (e.g., due to poor connectivity). Clearly, we cannot expect to guarantee reliable control and actuation when the connectivity to the Cloud is itself unreliable. Some representative examples of such situations can be found in the following industries: a) sensing the health of pipeline infrastructures in the oil and gas sectors (e.g., for detecting leaks); b) smart and precision agriculture; c) control systems inside certain factories, etc. The second niche gathers control systems and applications running in closed-loop form, which typically require very low latency. For instance, consider applications devoted to control and actuation in IoT (e.g., supported through M2M communications). While the actuators could be based on a relatively simple logic, the input space and data processing tasks could require considerable storage and computation-intensive operations before enforcing a control decision; and more importantly, control processes often demand upper bounds in terms of latency. In other words, the devices in charge of the sensing and actuation tasks often lack the computational power to assess the states and enforce the required control, and therefore, will need to rely on external compute and storage resources. However, current communications and Cloud infrastructures are not prepared to guarantee the deterministic latency bounds required by many control and actuation systems (e.g., in real-time control). As it can be foreseen, a smart combination of MCC and Hetnets can probably do the job and guarantee very low latency—at least for entirely fixed systems within pico or femto cells. However, as soon as we start considering applications with input sets that can grow randomly—especially when they can contain inputs produced by rapid mobility patterns—the whole idea of computing on the Cloud while ensuring bounded latency starts vanishing again.

## IV. Data Aggregation and Analytics in IoT

The set at the bottom of Fig. 1 covers another very important group of scenarios that typically involve data management, data processing, real-time analytics, and making very fast decisions based on the data analyzed. The centralized model offered by traditional Cloud facilities makes it perfectly suitable for controlling and managing data produced by a large number of applications. However, when many geographically distributed "things" produce data demanding analytics along with prompt decisions (e.g., under closed-loop control), hierarchical data processing models seem the only realistic alternative. Instead of sending "everything" to the Cloud, resources placed very close to the data producers can be used for local processing, data analysis, and fast decision making. In this way, relevant data can be aggregated and pushed to the next level in the hierarchy (e.g., to a Cloud facility), only when its content is important at a system-level for that particular application.

In general, the applications that require external compute and storage capacity in this group are characterized by combinations of the following factors: 1) large geographical footprint; 2) large scale (e.g., a large amount of sensors acquiring and sending data).; 3) large amount of data that need to be processed, aggregated, and exposed to other processes that consume these data; and based on the data processed offer 4) real-time analytics.

Observe that, in general terms, Cloud Computing could handle each of the four items listed above—though non-simultaneously. In fact, the first barrier for the Cloud becomes evident when we consider applications that have these four features and/or requirements together. The most robust and largely distributed systems that have been conceived thus far rarely have components that need data from entities that are far apart for performing their most basic and critical functions. It is at a system-level when cross-dependencies among entities that are far apart typically arise. We tend to build systems that tackle scalability by means of aggregation and hierarchy. This has very important consequences in terms of real-time analytics, since the semantics of the analysis will typically depend on the location (e.g., an event produces data that is very important for a local process, but it is often irrelevant at a global level). Note that this is particularly important in the framework of IoT and the deployment of Smart City applications.

In this context, and leaving aside rather obvious scalability issues, pushing all data to the Cloud will simply not be needed since sooner or later some sort of aggregation and localized analytics will be preferable for handling part of the job.

The second and most important barrier for the Cloud arises when, in addition to the requirements exposed above, we also consider fast mobility, and/or latency constraints, and/or poor or unreliable communications to the Cloud (see Fig. 1).

## V. Fog Computing

The main conclusion that can be drawn after our analysis is that Cloud Computing faces really complex challenges for supporting the requirements of each of the sets shown in Fig. 1 independently, but the true complexity lies in the intersections of the sets. A combination of Mobile Cloud Computing (MCC) and Hetnets will not suffice—at least not as they are conceived right now. Thus, the most natural and promising way to deal with these challenges is Fog Computing [1], [2].

The essence of Fog computing is schematically shown in Fig. 4. A comprehensive IoT platform will need to deal with six domains, namely: (1) the domain of the "things", composed by both fixed and mobile devices, sensors, etc., most of which can be characterized as either M2M or H2M devices and M2M gateways; (2) the network domain, covering the edge, the aggregation and the core; (3) the Cloud domain; (4) the service and application domains; (5) the users domain; and (6) the Fog nodes (i.e., the Fog-enabled elements providing the compute, storage and network capabilities to the "things"), which, as shown in the figure, will be scattered from the end devices right up to the Cloud. Each of these domains poses different requirements to the IoT platform, and will demand specific actions and treatment from the control and management layers.

Let us now analyze again the potential scenarios covered by the three sets shown in Fig. 1, in the context of an IoT platform capable of managing an infrastructure like the one depicted in Fig. 4. Clearly, if the "things" remain in a fixed position, the data processing can be offloaded to the Cloud, but as soon as the endpoints start moving—especially if they move fast—the Fog infrastructure will be in a much better position for processing the data. Indeed, virtual appliances can be outsourced to the Fog, and they can literally move with the endpoint through the Fog. For scenarios requiring high reliability and/or predictable latency, the goal should be to locate the intelligence where it is needed in the network (i.e., the compute and storage resources must be placed very close to the monitoring and actuation points), or even be embedded on them, thereby enabling higher reliability and localized closed-loop control. Finally, Fog enables: 1) data aggregation at the edge as well as pushing and pulling data selectively from the Cloud; 2) it can perfectly adapt and scale upon geographical expansions; 3) and more importantly, it scales much better than the Cloud for running data analytics and making real-time decisions on many different IoT domains.

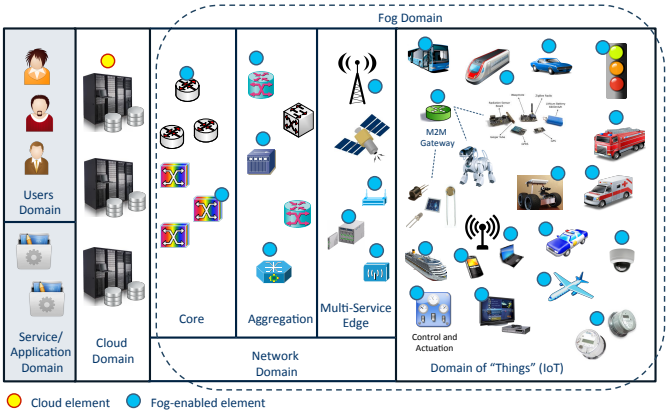



Fig. 4. The Fog Computing approach.

## VI. Conclusion

Compute and storage resources are becoming much more embedded in the network, and its footprint will eventually extend from the domain of the "things" at the edge of the network up to the Cloud without glitch. This trend offers remarkable opportunities, and almost naturally exposes the drivers for developing Fog Computing. More precisely, while the Cloud offers unprecedented efficiency, scalability, and cost reductions through virtualized IT infrastructures, these latter remain centralized in data centers, so the traditional Cloud model falls short in covering a series of growing needs in the IoT market. We have shown that a smart combination of Fog and Cloud Computing is the most plausible bet for building an adaptable and scalable platform for IoT. It is also worth noting that many verticals in IoT require the deployment of proprietary "boxes", and this is a serious problem in Smart City environments. This issue is another very important driver for Fog Computing, since the Fog naturally offers "box consolidation".

## Acknowledgment

This work was supported by the Spanish Ministry of Science and Innovation under contract TEC2012-34682 (partially funded by FEDER). The authors would also like to acknowledge the insight and fruitful discussions with J. Balaguer, X. Freixa, and I. Errando from Cisco Systems Barcelona.

## References

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. "Fog Computing and its Role in the Internet of Things". In *Proceedings of the ACM SIGCOMM 2012*, SIGCOMM '12. ACM, 2012.

[2] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu. "Fog Computing: A Platform for Internet of Things and Analytics". *Big Data and Internet of Things: A Roadmap for Smart Environments, Springer, Studies in Computational Intelligence*, 546:169–186, March 2014.

[3] L. Lei, Z. Zhong, K. Zheng, J. Chen, and H. Meng. "Challenges on Wireless Heterogeneous Networks for Mobile Cloud Computing". In *IEEE Wireless Communications*, June 2013.

[4] LXC - Linux Containers: https://linuxcontainers.org/.

[5] DOCKER: https://www.docker.io/.

[6] CRIU: http://criu.org.

[7] D. Farinacci, D. Lewis, D. Meyer, and V. Fuller, The Locator/ID Separation Protocol (LISP) IETF, RFC 6830, 2013.

[8] M. Handley, O. Bonaventure, C. Raiciu, and A. Ford, TCP Extensions for Multipath Operation with Multiple Addresses IETF, RFC 6824, 2013.