Received February 28, 2020, accepted March 18, 2020, date of publication April 1, 2020, date of current version April 21, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2984930

HOlistic pRocessing and NETworking (HORNET): An Integrated Solution for IoT-Based Fog Computing Services

PAOLO BELLAVISTA[1], (Senior Member, IEEE), CARLO GIANNELLI[2], (Member, IEEE), DMITRIJ DAVID PADALINO MONTENERO[1], (Student Member, IEEE), FILIPPO POLTRONIERI[3], (Student Member, IEEE), CESARE STEFANELLI[3], (Member, IEEE), AND MAURO TORTONESI[2], (Member, IEEE)
[1]Department of Computer Science and Engineering, University of Bologna, 40126 Bologna, Italy
[2]Department of Mathematics and Computer Science, University of Ferrara, 44121 Ferrara, Italy
[3]Department of Engineering, University of Ferrara, 44122 Ferrara, Italy

Corresponding author: Mauro Tortonesi (mauro.tortonesi@unife.it)

ABSTRACT Fog Computing is a recent and compelling paradigm that proposes to run information-processing services at the edge of the network. While interesting standardization efforts in Fog Computing are being currently pursued by many organizations, most of them focus on management and orchestration functions, and primarily propose the adoption and adaptation of programming models designed for Cloud applications. Instead, Fog Computing applications would significantly benefit from innovative solutions that, on the one hand, adopt an "acceptable lossiness" perspective and manage information processing/dissemination in a dynamic and integrated way and, on the other hand, support a Multi Layer Routing (MLR) approach to exploit multiple routing options at different abstraction levels at the same time. This paper presents an overview of the opportunities and challenges of Fog Computing-based Internet of Things (IoT) applications by jointly exploiting acceptable lossiness and MLR. In addition, the paper proposes the innovative Holistic pRocessing and NETworking (HORNET) Software Defined Networking (SDN) solution which leverages an information-centric and value-based service model and the MLR approach to support IoT applications. The reported preliminary experimental results show the feasibility and effectiveness of the proposed approach.

INDEX TERMS Fog computing, IoT, network softwarization, value-of-information (VoI).

I. INTRODUCTION

Fog Computing is a recently emerged paradigm that allocates information-processing services at the edge of the network, that is, on top of edge devices in proximity of raw data sources, information consumers, or both [1], [2]. Fog Computing is of significant relevance for Internet of Things (IoT) applications, as it reduces response latency (and consequently improves the quality) of IT services and alleviates the burden on the network infrastructure [3]–[5].

While several related standards have been recently proposed [6], Fog Computing still remains a rapidly evolving industrial and academic research topic. Several proposals focus on concurrent solutions managing resources at the information processing and communication layers. Among those, some relevant works investigate resource allocation [7], [8], QoS-aware application deployment [9], [10], and network softwarization issues at the Software Defined Networking (SDN) [11], [12] and slicing [13] levels.

Most of those proposals adopt an approach that extends to Fog environments concepts that were developed for Cloud ones, such as application programming models and QoS paradigms. Instead, we believe that Fog Computing applications would significantly benefit from innovative solutions specifically designed to consider the fundamental aspects of that environment, such as the time-sensitive, location-aware, and information-centric nature of IT services and the scarcity of resources. In this context, approaches that jointly address

The associate editor coordinating the review of this manuscript and approving it for publication was Honghao Gao.

the issues of information prioritization for processing and dissemination and of traffic engineering seem particularly well suited to address the intrinsic dynamicity of Fog Computing scenarios.

This paper investigates the aforementioned challenging but compelling research avenue and proposes the *HOlistic pRocessing and NETworking (HORNET)* solution.

HORNET adopts an innovative holistic SDN-based approach that simultaneously and jointly considers both transport (for packet re-routing and engineering) and application (for dynamic deployment and de/activation of services) layers. This represents a significant difference from traditional SDN approaches focusing only on the networking perspective and allows HORNET to perform an optimal management of both computational resources and network flows configuration.

Within this general approach, HORNET adopts a service model which was specifically designed to address the requirements and challenges of Fog computing applications. First, the HORNET service model follows an ''acceptable lossiness'' perspective to Fog service development, at both the processing and communication levels. The assumption is that Fog environments are resource scarce and might not be able to fully support application requirements. For this reason, services should focus their effort on the processing of a subset of data that allows them to maintain acceptable Quality of Experience (QoE) levels, eventually dropping packets related to data deemed as less crucial.

In addition, HORNET Fog services are information-centric, adaptive, and realized on top of service components instantiated along the communication path from (raw) information producers to (processed) information consumers. Service components are loosely-coupled and composition-friendly software modules that can be quickly chained to create applications, and just as quickly rearranged in case of need. This paradigm is inspired by research on SDN and Network Function Virtualization (NFV), and extends the service chaining concept as a composition of different Virtual Network Functions (VNFs) to a higher-level application-driven perspective, by considering information-centric service components instead of VNFs. In addition, HORNET addresses the dynamicity of the application scenario by implementing at the single service component level a processing logic which is adaptive and capable of reducing requirements to match the level of resources currently available in the deployment location, i.e., the edge device on which the component is running.

To dynamically control and optimize information dissemination between service components, HORNET also introduces an important innovation. More specifically, HORNET leverages Multi Layer Routing (MLR), which provides three different forwarding dissemination tools to tackle the needs of different types of applications. MLR dynamically uses multiple routing solutions at different abstraction layers, ranging from native IP ones to more expressive but resource-consuming packet dispatching techniques, also considering the actual content within packet payload. The adoption of MLR allows HORNET to tailor the specific dissemination solution for processed information according to service requirements and the current execution context, by dynamically choosing the most suitable among a wide array of different options.

Finally, HORNET adopts Value-of-Information (VoI) as a unifying criterion for the optimization of resource allocation and management. Originally born as an extension of Shannon's information theory in the 1960s for decision making purposes, the VoI concept has been recently attracting interest in scientific literature on tactical and Fog computing applications as a subjective metric that measures the utility that Information Objects (IOs) deliver to end recipients [14], [15]. Compared to other criteria, VoI naturally enables a much more effective usage of the scarce and heterogeneous computational and bandwidth resources in information processing and dissemination.

The adoption of these innovative models provides to HORNET several advantages compared with traditional solutions. It allows to achieve the desired Quality of Experience (QoE) levels by focusing on the processing and dissemination of the most valuable pieces of information from the end users' point of view, possibly delaying or even dropping less valuable ones.

However, since it is a radical change of paradigm to take advantage of the HORNET platform each service would have to be developed in accordance with HORNET concepts and API, thus allowing a VoI-based management of both network flows and service components. To address this issue, we explicitly consider backwards compatibility in the design of HORNET. We envision that HORNET can be adopted either in full as a comprehensive architecture or embedded into an existing system. In the latter case, HORNET would manage the behaviour of existing services and solutions by means of more traditional methodologies such as native IP.

This paper extends our previous conference work [16] about the HAN middleware. The proposed HORNET solution introduces significant improvements at the architectural and implementation levels over HAN. Specifically, to develop effective solutions able to tackle the complexity and dynamicity of Fog Computing applications, we put our efforts in integrating the VoI concept inside both the management logic of the control plane and the dissemination data plane, thus allowing to prioritize the processing and dissemination of most valuable application-specific information.

The paper also presents a thorough experimental evaluation of the HORNET solution, both in a real testbed and in a realistic simulated environment. The results show that HORNET is capable of quasi real-time network layer reconfigurations, even on resource limited hardware, and validate the HORNET VoI-based resource allocation approach by demonstrating its effectiveness in a dynamic context with multiple concurrent services.

II. RELATED WORK

While the SDN approach has been traditionally adopted (and by now it is considered the standard solution) in datacenters and huge enterprise networks [17], some first research efforts demonstrate its validity also in Fog environments [12], [18]–[21]. To this purpose, the section revises state-of-the-art contributions along three primary directions: i) the current literature proposing to adopt the SDN approach in Fog environments, ii) the recent trend towards softwarization of network services taking advantage of the SDN, and iii) dynamic service composition and task offloading to better manage Fog nodes and related computational/resources, eventually also adopting Quality-of-Information (QoI) and Value-of-Information (VoI) concepts.

As presented in a recent survey [22], most of SDN-based Fog routing solutions propose the adoption of SDN technologies in Fog Computing to provide efficient routing mechanisms capable of addressing low latency, low bandwidth, and security requirements of Fog Computing environments. In addition, the paper presents an SDN solution based on a hierarchy of Fog controllers distributed between the Cloud and the edge; the primary idea is that frequent events are locally managed on Fog controllers and rare events are globally managed by the Cloud controller. Other papers focus on security aspects related to the adoption of SDN in Fog Computing. For instance, [23] analyzes security vulnerabilities of the OpenFlow channels and proposes an attack model and related countermeasure based on Bloom filters. Moreover, [12] proposes a novel approach based on the combination of SDN and Blockchain to securely manage computational resources (Fog nodes) at the edge, with the primary goal of exploiting the Blockchain to connect together multiple SDN controllers deployed in a distributed manner in the Fog layer. Finally, [24] outlines how another relevant application of SDN in Fog Computing can be the identification of threats and anomalies. In particular, the paper exploits the SDN control plane to dynamically deploy software agents monitoring the traffic to identify anomalies and eventual attacks.

The wide adoption of SDN together with Network Function Virtualization (NFV) has more recently pushed the attention on the softwarization of network services traditionally implemented in hardware [25], [26]. In this scope, Service Function Chaining (SFC) aims at providing an efficient composition and/or orchestration of different and related network applications/functions to achieve a chain of services (building block services) suitable for devices and users in Fog Computing environments. In [27], Yu *et al.* propose the adoption of SDN as the enabling methodology for QoS traffic steering in SFC. More specifically, the authors give a mathematical definition and formulate a polynomial time approximation algorithm. Instead, Zhang *et al.* discuss rule management in SDN-based IoT and propose to aggregate and minimize the number of forwarding rules by multiplexing different traffic flows flowing in the same path aggregating them with a VLAN ID label [11]. In [28], the authors describe a network function virtualization-aware orchestration for SFC placement in the Cloud. In particular, the authors propose an heuristic for component orchestration in small- and large-scale DC network (BACON) that minimizes the intra- and end-to-end latency of the SFC. A similar approach is given in [29], where authors formulate an optimization problem for the deployment of service function chain in 5G mobile networks. Furthermore, [30] presents an interesting survey focusing on Fog Computing applications, pointing out some of the current challenges that orchestration techniques usually find in Fog Computing scenarios: churn and unreliability of nodes at the edge, heterogeneity of resources, security and privacy related issues, and location issues.

Finally, some work focus on dynamic service deployment [31] and data management optimization in [32] in IoT environments. For instance, [33] proposes Data-intensive Service Edge deployment scheme based on Genetic Algorithm (DSEGA) to identify a perfect fit for component services and data deployment on edge nodes in relation to storage constraints and load balancing conditions. Instead, [34] presents a work offload solution considering the geographic location of mobile edge servers and IoT devices to better serve service subscribers, also to decide if and when a task should be run locally or on a remote edge node. Moreover, [35] introduces an innovative approach to predict the QoS in IoT environments based on Neural Collaborative Filtering (NCF) and fuzzy clustering. The main idea is to cluster contextual information and exploit a new combined similarity computation method to identify latent features in historical QoS data.

More recently the QoI and VoI concepts are starting to emerge in the literature [36]. In fact, the adoption of VoI-based metrics could enable information and flow prioritization at multiple levels and abstraction layers, and represents a particularly interesting research avenue for IoT applications [37]. Bellavista *et al.* propose VoI as criterion to develop a sensor service ranking mechanism called VoISRAM in [38]. The main goal is to exploit the ranking mechanism gateway services to achieve a trade-off between application specific QoS requirements and the network energy consumption. Bisdikian *et al.* present an application-agnostic QoI specification to enable the assessment of information quality across different applications. In [39], the authors propose a distributed algorithm for data collection called EQRoute that uses QoI criterion to maximize information value and reduce energy consumption.

In conclusion, we claim that the SDN adoption on the edge side of Fog environments requires more efforts since there is the need to adopt a wider and holistic point of view. The objective is to support Fog service reconfiguration not only considering networking features and resources (such as most of the current literature propose) but also service composition based on the aggregation of processed data (as we originally propose in this paper).

III. REFERENCE SCENARIO

To better illustrate how HORNET addresses the issues of Fog computing applications, let us consider a Smart City scenario hosting several services: pollution monitoring, traffic monitoring, and plate recognition. Pollution monitoring analyzes environmental data collected from, e.g., CO_2, NO_x, humidity, and temperature sensors to produce a report on pollution levels, then disseminated to citizens in the area. Traffic monitoring analyzes data from traffic cameras and vehicle counting sensors to evaluate the level of traffic congestion in the city; then, it disseminates situation reports to citizens (either pedestrians or drivers) and police officers in the surrounding area. Finally, the plate recognition service monitors the traffic plates, e.g., to make sure that only cars with a specific permit are driving through restricted areas. Cars not complying with this restriction are automatically reported to the police.

All these services have a long-time/continuous running nature and will thus leverage software components running on Fog nodes deployed and administrated by the municipality. The fair allocation of resources to this set of various and ever-running services represents a challenge. In fact, static allocation of resources, simply based on weights assigned to service types, would be inadequate in this scenario. In fact, in steady state, all the above mentioned services will likely have access to all the resources they require. But this allocation would not make the system responsive to quick changes of the environment context.

For instance, consider the case of a stolen car report, leading to the insertion of a car plate code to look for in the plate recognition service with high priority. In this case, the plate recognition service becomes significantly more important and the VoI of its associated traffic flows increases considerably. Our SDN-based HORNET solution dynamically manages the transport layer of the Fog environment by configuring how traffic flows with higher VoI are dispatched, i.e., by tuning routing tables to reroute such flows towards less loaded links, thus prioritizing relevant flows promoting them from regular data streams (whose messages may be delayed or eventually dropped by adopting an ‘‘acceptable lossiness’’ approach) to high-priority data streams (that have full access to networking resources of municipality Fog nodes), or both.

Furthermore, consider the case of a missing child report, requiring to analyze a huge amount of data (dramatically greater than in the plate recognition service) collected from cameras by applying computationally expensive face recognition algorithms. The effective and efficient provisioning of such a service would require not only the prioritization of associated traffic flows (like in the previous example), but also (and most relevant) the on demand deployment and instantiation of service components close to raw data sources, i.e., cameras. To this purpose, our SDN-based HORNET solution adopts a two-layer approach: at the application layer it selects if, where, and when there is the need of instantiating novel service components, while at the transport layer it prioritizes messages carrying the new service (either its configuration or the service component itself) to quickly activate it. Let us note that by applying face recognition algorithms in newly deployed fog devices close to the cameras it is possible to achieve the notable twofold benefit of increasing the scalability of the service (thus reducing its latency) and of decreasing the traffic on the network (since video streams can be locally processed instead of dispatched to a central server running on a Cloud platform).

To this end, traditional management solutions, or even recent SDN ones leveraging the concept of network slicing, are not enough to address all the issues raised by the above challenging scenarios. Therefore, to tackle the impelling requirements of such scenarios there is the need to develop solutions capable of reallocating networking and computational resources to service components in a rapid, fair, and fine-grained manner, while addressing the issue of modulating the information dissemination substrate to match the current state and application requirements.

IV. THE HOLISTIC pRocessing AND NETworking (HORNET) APPROACH

We devised the *HOlistic pRocessing and NETworking (HORNET)* SDN-based solution to address the issues discussed in the previous reference scenario. HORNET integrates and extends our past work on the Real Ad-hoc Multi-hop Peer-to-peer (RAMP) [40] and the Sieve, Process, and Forward (SPF) middleware [41], by dynamically managing the deployment and composition of Fog services according to VoI optimization criteria and by dynamically re-configuring end-to-end communications to maximize QoS [42], [43].

As illustrated in Fig. 1, HORNET executes its middleware services on top of edge devices (nodes N1-N5 in the figure) capable of hosting information processing tasks and of operating as routers for traversing flows, coordinated by a centralized Fog SDN Controller. At each node, HORNET instantiates and manages information processing and dissemination tasks in an integrated and context-aware way by leveraging a two-layer approach, with a top service layer and a bottom communications layer.

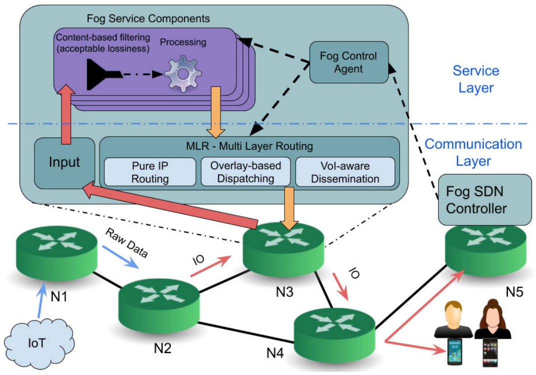

FIGURE 1. The HORNET Concept.

At the *service layer*, the Fog SDN Controller dynamically re/deploys and de/activates service components on fog/edge devices. Service components can be any software module, ranging from shell scripts and OSGi bundles to containers and even virtual machines, of course with different supported capabilities and different performance in terms of deployment latency. To this purpose, the Fog SDN Controller analyzes service requirements and the generated VoI to modulate resource allocation, by possibly deploying missing HORNET components when and where needed. At the *communication layer*, the Fog SDN Controller dynamically manages Fog nodes to improve the QoS of packet dispatching not only by computing best paths towards the destination and by tuning forwarding rules on intermediate nodes, but also by adopting an SDN-based MLR approach to select the most proper routing mechanism among different abstraction layers.

A. SDN-BASED MULTI LAYER ROUTING

We believe that to support efficient deployment and integration of information-centric services with highly varying fan-in and fan-out there is the need of a communication layer capable of implementing a flexible and context-aware network fabric. To this end, HORNET takes advantage of the MLR-based data plane (in conjunction with the RAMP middleware) supporting three primary routing mechanisms at increasing expressiveness power: pure IP forwarding, (VoI-oblivious) overlay-based dissemination, and VoI-aware overlay-based dissemination.

Pure IP forwarding works by dynamically modifying per-device routing tables with traditional iptables command to reroute vanilla UDP and TCP traffic flows based on IP address destination. This mechanism manages legacy applications in a completely transparent way for edge devices, thus simplifying QoS management.

Overlay-based dissemination supports collaborative packet dispatching among edge devices by adopting routing schemes based on flow ids. In this case, the Fog SDN Controller provides a flow id to applications to label generated traffic with. In addition, the Fog SDN Controller manages Fog nodes by specifying, e.g., that the traffic labeled with a given flow id has higher priority and thus other lower-priority traffic flows should be temporarily delayed.

VoI-aware overlay-based forwarding extends the previous mechanism allowing to tune packet forwarding not only based on the flow id, but also on the VoI carried by each packet. To this purpose, the Fog SDN Controller manages edge nodes by offering routing rules based on VoI values. Let us note that in this manner it is possible to support VoI-dependent information dissemination, e.g., by specifying that for a given traffic flow packets carrying data with VoI values below a threshold should be discarded, or differentiating destination nodes based on VoI ranges at the sender as well as on intermediary nodes. In other words, this mechanism dynamically modifies the pipeline of fog services by also permitting the definition of different pipelines for the same data, based on time-varying VoI values. Furthermore, the VoI-aware forwarding also enables a prioritized delivery of service components to Fog nodes, thus allowing a fast instantiation of high VoI service components even in case of a network congestion.

The Fog SDN Controller selectively adopts one or more of the above MLR mechanisms, with per-application granularity, depending on the current state of the network and on the specific application needs. To this end, the Fog SDN Controller would likely adopt Pure IP forwarding or Overlay-based forwarding for legacy applications, while applications built on the top of VoI would adopt VoI-aware overlay-based forwarding. For instance, the Fog SDN Controller can decide that a legacy application with strict low latency requirements (such as video streaming for face recognition [44] or image captioning [45]) should exploit pure IP forwarding, with no overhead due to overlay networking. On the contrary, more articulated applications could be effectively provided only if supported by additional dynamically deployed components, eventually coupled with value-dependent packet dissemination techniques, e.g., by disseminating plate recognition information in a differentiated manner to prioritize newly stolen vehicles.

Let us note that while the data plane is based on and enabled by the MLR approach (thus IP native, overlay networking, and VoI-aware dispatching), the control plane is based on the overlay network only. In this manner, information and commands sent to Fog nodes and the Fog SDN Controller can take advantage of the routing flexibility of overlay networking. For instance, in this way it is possible to identify destinations based on a network-independent unique node id rather than an IP address that could change or could be duplicated in different subnets of the same multi-hop Fog environment. In other words, the Fog SDN Controller can dispatch packets related to the control plane independently of how (and whether) routing tables on intermediary Fog nodes have been configured.

The reference scenario presented in Section III can greatly benefit from the MLR approach adopted by HORNET. For instance, the Fog SDN Controller can exploit MLR to configure the network to reroute low-priority pollution traffic towards limited bandwidth longer paths supported by the overlay network. On the contrary, it can setup IP routing tables on intermediary Fog nodes to forward high-priority video streams for face recognition towards IP-based short paths.

B. VALUE-BASED INFORMATION PROCESSING

The HORNET solution adopts the MLR approach and extends it with the *Adaptive, Information-centric, and Value-based (AIV)* service and information maturity model [46]. On the one hand, AIV proposes an information maturity model that classifies messages in two different categories: raw-data if a message is generated by a sensor and Information Object (IO) if the message is the processing result of a service component. On the other hand, AIV proposes an innovative Fog service model that leverages VoI-based concepts to enable the development of services capable of automatically scaling their resource requirements to their current execution context while preserving high QoE levels.

More specifically, AIV assumes that the processing function of a Fog service emerges as the result of the coordinated orchestration of adaptive and composition-friendly service components, in which each component is responsible to deal with a part of the information processing. This loose definition of Fog services allows to easily support dynamic architectures where single instances of service components can be migrated to different devices along the Cloud-IoT continuum according to the current execution context (service requirements, resource availability, user preferences, etc.). Above this concept, Fog services are defined as a topology of service components connected together with respect to a service description that defines the semantics, the characteristics, and the interactions between service components. For example, the face recognition service mentioned in the reference scenario implements the processing of video feeds collected from nearby cameras (raw-data) through several phases: video transcoding, face detection, and face recognition processing. In this case, a first service component takes care of video transcoding operations by transforming the cameras frame raw-data into IOs that will feed the face detection service component, which in turn passes its output IOs to the face recognition service component to produce valuable IOs for end users interested in consuming them.

The development of VoI-aware Fog services requires explicit support at the middleware level. In fact, there is the need of VoI evaluation mechanisms of raw data and IOs that are of generic applicability and can be configured to match the needs of the specific service components where the VoI evaluation is used. To this end, we developed the SPF middleware, which aims to be the reference implementation of the AIV service model, and released it as open source at: `https://github.com/DSG-UniFE/spf` [41].

SPF evaluates the VoI of messages and keeps track of it along their lifecycle. More specifically, the initial VoI associated to an IO m is calculated as a function of the messages processed for the IO generation and of the priority of the originating service component:

$$VoI_0(m) = SSV(\mathbb{I}(m)) \times FSP(sc(m)) \tag{1}$$

where $\mathbb{I}(m)$ is the set of input messages processed for the generation of IO m and $sc(m)$ is the service component that generated m. (Note that $\mathbb{I}(m)$ will typically consist of raw data messages, but might also include IOs, as it is not rare for IoT services to perform information processing at different abstraction levels.) $SSV(\mathbb{I}(m))$, as in "Service Specific Value (calculation)", is a factor that takes into account service-specific considerations when assessing the value of the information extracted from $\mathbb{I}(m)$. $FSP(sc(m))$, as in "Fog Service Priority", is a factor that considers the priority of the service that component sc belongs to, thus assigning higher VoI to the IOs produced by higher priority services.

The result of equation (1) represents the basis to calculate the VoI that message m delivered to its recipients. More specifically, m needs to consider other elements such as the decrease in the value for time sensitive information and the decrease in the value for location-aware information. We then have:

$$VoI(m, r) = VoI_0(m) \times TRD(r^t, m^{ot}) \times PRD(r^l, m^{ol}) \tag{2}$$

where $TRD(r^t, m^{ot})$, as in "Timeliness Relevance (of Request) Decay", is a factor that takes into account the loss of VoI in the time elapsed between message generation m^{ot} and receival r^t and, $PRD(r^l, m^{ol})$, as in "Proximity Relevance (of Request) Decay", is a factor that considers the loss of VoI as it traveled from the originating location m^{ol} to its recepient location r^l.

The total VoI delivered by an IO message m then becomes:

$$VoI(m, \mathbb{MR}(m)) = VoI_0(m) \times \sum_{r \in \mathbb{MR}(m)} [TRD(r^t, m^{ot}) \times PRD(r^l, m^{ol})] \tag{3}$$

where $\mathbb{MR}$ is the (set of) receivers for message m. As it can be seen, the total VoI produced by a service can be high either because the messages it generates are dispatched to a considerable amount of users or because associated service components are providing highly valuable IOs. For more details about VoI tracking, both within SPF and as a general framework, we kindly refer the reader to [15].

At the single node level, SPF keeps track of the total VoI of the IOs generated by the service components running in that node. It then uses this information to assign local (computation, storage, and bandwidth) resources to service components according to the VoI they generate.

C. OPTIMAL PROCESSING AND NETWORKING CONFIGURATION

HORNET builds on top of SPF to run VoI-based information processing services and leverages its VoI tracking capabilities to tailor the communication layer according to the application requirements and the current VoI they are producing. More specifically, HORNET aims at finding the service component allocation α and network configuration γ which optimize the total VoI delivered to end users:

$$\underset{\alpha,\gamma}{argmax} \sum_{m \in \mathbb{M}(t_n, t_{n+1})} VoI(m, \mathbb{MR}(m)) \tag{4}$$

where $\mathbb{M}(t_n, t_{n+1})$ are the messages received by end users within the (t_n, t_{n+1}) time window.

To optimize service component allocation and traffic engineering, HORNET needs up-to-date information about the VoI delivered by Fog services. To this end, SPF continuously monitors service components and, through the local Fog Control Agent, periodically informs the Fog SDN Controller of the total VoI of generated raw data and IOs and if they require additional resources. By using the total VoI associated to raw data and IOs as a resource assignment criterion, the Fog SDN Controller selects the best path for traffic flows and prioritizes the assignment of resources to services that are providing the highest VoI value to their end users. Furthermore, whenever the Fog SDN assesses that rerouting and prioritization together do not achieve the required QoS (or the generated VoI is extraordinarily high), it can also select to re/deploy new service components.

HORNET uses its knowledge base about the services currently running in the network and the VoI tracking information collected from SPF to optimize service component allocation and traffic engineering at the entire system level. More specifically, HORNET solves the holistic optimization problem in equation (4) by leveraging a continuous optimization solution based on an advanced genetic algorithm variant with adaptive mutation. Genetic algorithms are particularly well suited for optimization problems with complex search spaces, because of their remarkable flexibility in chromosome representation, and dynamic aspects, which they are capable of addressing effectively using a varying intensity mutation process (controlled by feedback on convergence speed) [47] and hypermutation triggering to deal with abrupt changes in the system state [48]. As a result, genetic algorithms represent a very good choice for HORNET. However, genetic algorithms are known to suffer from relatively slow convergence rate in some cases. To address that issue, we are also exploring alternative optimization solutions leveraging Quantum-based Particle Swarm Optimisation and greedy algorithms.

Let us clarify that the VoI function in equation 4 represents a scalar/univariate quantity. This allows to formulate the resource assignment problem as a single-objective optimization one, and actually represents a significant advantage of the adoption of VoI as an underlying theoretical framework. Alternatively, non VoI-based multi-objective formulations of the resource allocation optimization problems would have required the adoption of a significantly more sophisticated optimization solution, for instance NSGA-II, and most likely less performing from the convergence rate perspective.

The holistic approach to information processing and traffic engineering management adopted by HORNET allows to effectively maximize the overall QoS of the most important service components, i.e., those producing the highest VoI. For example, in case of a stolen vehicle, information about car plates suddenly becomes much more important – hence valuable – for the police. As a result, since the plate recognition service starts delivering higher VoI, the HORNET SDN controller exploits MLR mechanisms at the transport layer by increasing the priority level of packets carrying information about car plates. Once notified about the new priority level, intermediary nodes start dispatching such packets with higher priority as soon as they arrive, but only if the per-packet VoI is high. For instance, a packet carrying a car plate whose picture is blurred (i.e., not useful for the service) has a high priority but low VoI, and thus it can be delayed. On the contrary, intermediary nodes forward packets related to other services with lower priority only if and when there are no plate packets in the queue. In other words, the dispatching of packets with lower priority is delayed whenever a car plate packet arrives; in case the queue of outgoing lower VoI packets increases too much (and thus packets are delayed for a long period), eventually they can be dropped in an ‘‘acceptable lossiness’’ fashion.

This is even more relevant in the case of missing child report. Since it generates IOs with a very high VoI, the SDN controller exploits the application layer mechanisms to trigger the deployment of new service components in charge of applying face recognition algorithms close to cameras, thus providing more computational resources to process most relevant raw-data. To this purpose, it exploits MLR to provide maximum priority to packets carrying software components to be deployed, temporarily delaying the dispatching of traffic flows related to any service despite their priority and VoI. In this manner, it is possible to achieve the notable benefit of promptly delivering service components also in case of congested network. Then, if the child missing alarm is called off, the Fog SDN Controller sets priority to former levels and stop/decommission service components to release computing and networking resources.

Finally, let us stress that the VoI-aware MLR approach allows HORNET to decide how to optimize the communication layer during service provisioning, with no need to impose service stops for static reconfigurations at service launch time. For instance, the traffic monitoring service typically has a high number of consumers and can be efficiently carried on top of the overlay based dissemination mechanism, also taking advantage of device-to-device (D2D) communications and step-wise efficient multicasting. Furthermore, HORNET can transfer high-VoI video streams for face recognition over high quality paths without any delay and without dropping any packet, while assigning limited network resources to low-VoI video streams for plate recognition that can afford to be slightly delayed or partially dropped (‘‘acceptable lossiness’’).

D. ARCHITECTURE AND IMPLEMENTATION INSIGHTS

Fig. 2 outlines the overall architecture of a Fog service node in HORNET. The depicted components are deployed and activated on each Fog node and allow the node participation to the HORNET Fog environment. In addition, for each Fog environment there is one node acting as the Fog SDN Controller by registering itself to the local RAMP as ‘‘Fog SDN Controller’’ service, thus allowing remote Fog nodes to dynamically discover and register to it. Based on the information provided by remote Fog nodes, the Fog SDN Controller generates a weighted graph representation of the topology.

Delving into finer details of each Fog node, the overall architecture is divided into the Control Plane and the Data Plane. The Control Plane (Fig. 2, top) primarily consists of Link Connectivity Manager (LCM) and Control Agent (CA). LCM manages single-hop links and provides network status information. CA gathers information and exploits the overlay network to send data to the SDN controller and receive commands from the SDN controller. More specifically,

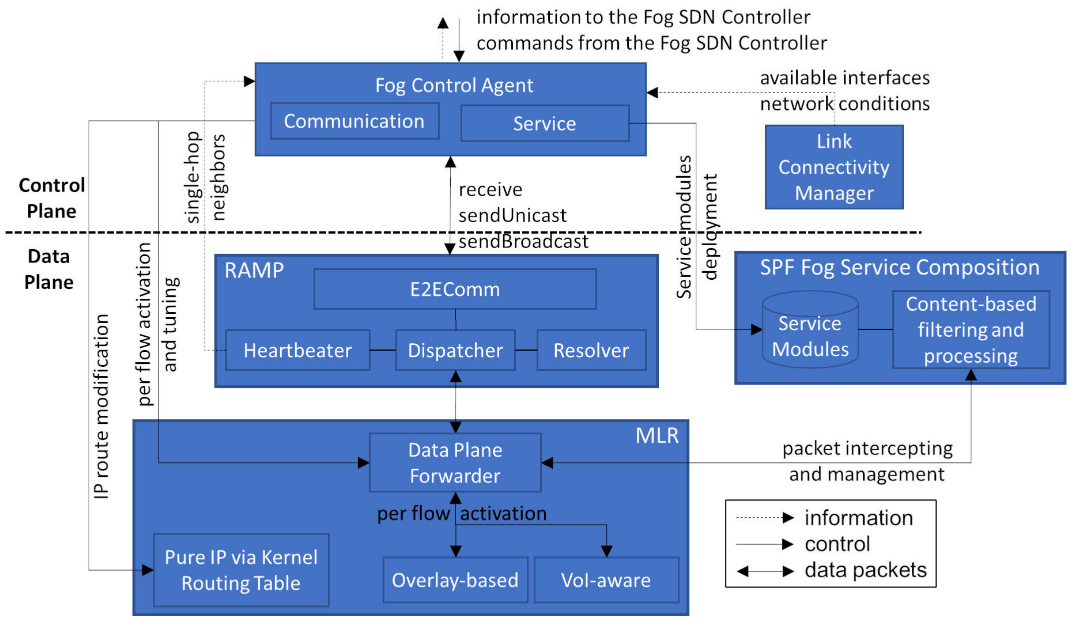

FIGURE 2. Architecture of the HORNET SDN solution.

the Communication sub-component appropriately controls the underlying MLR component to dynamically configure available routing mechanisms, while the Service sub-component dynamically deploys and activates software modules to enrich Fog nodes with additional capabilities required to correctly provide requested services.

The Data Plane (Fig. 2, bottom) consists of the RAMP middleware, the MLR component, and the SPF middleware enhanced for Fog service dynamic composition. The RAMP middleware supports the creation of the multi-hop overlay network with best-effort packet dispatching. MLR properly manages packets received by the RAMP middleware on a per-flow basis, by adopting the listener-based Data Plane Forwarder (DPF) to intercept incoming overlay network data packets and apply routing rules related to overlay-based and VoI-aware MLR layers. Furthermore, it is worth noting that Pure IP is logically part of the MLR component (and it is configured by the local SDN CA), but packet forwarding is actually performed by the operating system through kernel routing tables based on the received HORNET indications.

The MLR layer extracts the content (together with VoI metadata if available) of incoming packets of interest to any of the services running on the Fog node and forwards it to SPF, which in turn dispatches it to the concerned service component(s). In the (likely) case the processing leads to the generation of higher level IOs, the latter – along with their VoI metadata – will be forwarded to MLR in charge of selecting the proper communication mechanism and finally dispatching it.

The current implementation of the Fog SDN Controller adopts the Graph Stream library to identify best paths to provide Fog service, e.g., by applying Breadth First or Dijkstra's algorithms based on different cost functions. In particular, when an application requires to the Fog SDN Controller the best path to access a service, it also gets one of the three already developed routing mechanisms: Pure IP, i.e., managing the Fog environment to modify operating system routing tables of intermediary nodes; Overlay-based dispatching, i.e., exploiting the RAMP middleware to forward packets based on a flow id senders have to tag transmitted packets with; and VoI-aware, identifying the path towards the destination based on the dynamically calculated VoI of packets, e.g., to exploit large bandwidth and small latency for packets with high VoI (thus ensuring better QoE) and less capable paths for packets with low VoI.

V. EXPERIMENTAL EVALUATION

We have evaluated the proposed solution with a Java prototype as well as in a simulated scenario, with the twofold objective of demonstrating i) its feasibility and efficiency in a real-world (but small-scale) scenario and ii) its capability of dynamically deploy and and compose service components in a wider simulated environment.

In particular, in-the-field experiments (based on the RAMP middleware, see Section V-A) allow to evaluate the performance of MLR-based network reconfiguration by measuring the control plane latency, also considering the challenging case of service component deployment with bandwidth saturation due to high traffic load. Then, performance results achieved with the prototype are used to configure a larger-scale simulated environment (based on the Phileas simulator, see Section V-B). In this manner, we are able to faithfully reenact an actual prototype in a simulated environment to evaluate how the proposed solution is able to promptly reconfigure service components (and the network in general) by considering the activation of multiple devices and heterogeneous service instances to optimize the overall VoI.

A. IN-THE-FIELD PERFORMANCE EVALUATION OVER REAL TESTBED

Let us preliminarily notice that most related papers in the existing literature do not include experimentation over real testbeds and deployment environments, due to the time-consuming effort that this requires. On the opposite, we claim the relevance of such experimentation to collect feedback from the experience of real deployment and to determine real setting values for the configuration of associated simulations. To that purpose, we have built a testbed of 5 Raspberry Pi 3 Model B+ connected one another in a kite-like network topology via either Ethernet or IEEE 802.11 and based on our Java prototype available at `https://github.com/DSG-UniFE/ramp`.

We have evaluated the efficiency of the control plane along two primary guidelines: the time required to setup MLR forwarding rules on Fog nodes and the capability of dynamically deploying service components.

On the one hand, we have evaluated the overhead due to rule management by measuring the *rule management cost*, i.e., the time spent from when a node requires to the SDN controller the deployment of a new traffic rule to when the new traffic rule starts to be enforced on Fog nodes. In the case of pure IP, the SDN controller has to setup a new routing rule and send it to a Fog node, in charge of applying it. In case of VoI-aware forwarding, the SDN controller prepares the Java class containing the logic of the rule and sends it to a Fog node, deploying and registering it to the overlay networking RAMP middleware (we also tested the overlay-based dissemination case, with performance results that are very similar to the VoI-aware one).

To this end, Fig. 3 depicts measured traffic rule deployment costs in a regular situation, i.e., without network congestion. In the (native) IP case it takes about 483 ms, while in the VoI-aware case only 59 ms. Such a difference is justified by the fact that in the former case there is the need of modifying routing rules at the operating system layer, which is a time-consuming operation also due to the required context switch and the time required to execute the `iproute2` command. Instead, in the latter case, once the VoI-aware routing rule reaches the interested Fog nodes, the deployment and registration procedures take less time since they are implemented within HORNET directly at the overlay layer (they do not require reconfiguration at the operating system layer). In fact, at the reception of a file representing the rule as a Java class, HORNET stores the file locally, then instantiates it by directly interacting with the Java class loader (specialized to specify custom source directories specifically containing VoI-aware routing rules), and finally registers it in a key-value store with the available VoI-aware rules. Such procedure does not involve any time-consuming operation and can be efficiently done in much less than 100 ms as reported in Fig. 3.

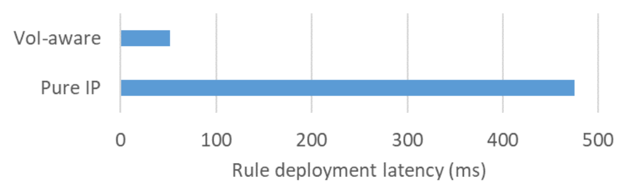

FIGURE 3. Rule management cost.

On the other hand, Fig. 4 shows the qualitative trend related to the delivery throughput of a service component (size of 5 MB) via a congested link (nominal bandwidth of 3000 KB/s). More specifically, at time 1 s a traffic flow carrying data packets starts to fully exploit the available bandwidth. Then, at time 7 s the SDN controller exploits the same link to deliver the service component at maximum priority. Slightly after the new control traffic flow starts, the previous one is inhibited by delaying the dispatching of its packets. Finally, at time 9 s the service component is completely delivered and networking resources provided again to the previous flow. In other words, the proposed solution is able to deliver control messages even if data flows saturate the bandwidth.

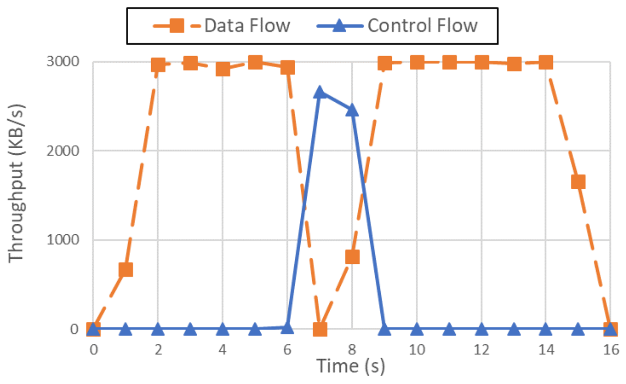

FIGURE 4. Priority management of control and data flows.

Finally, we have evaluated the *rule enforcement cost* metric, i.e., the time required to enforce rules on intermediary nodes. In this manner, we can better assess the suitability of the proposed solution for challenging use cases requiring the prompt dispatching of packets. To this purpose, we compared the time required on a Fog node to identify the next node without and with considering the VoI. In the former case, it requires to retrieve the flow id from the packet header and look up the overlay network routing table maintaining `<flow id, next hop>` mappings. In the latter case, there is the non-negligible additional overhead due to payload deserialization, required to retrieve the packet content and then compute the VoI.

Fig. 5 depicts how the rule enforcement value varies while increasing the payload size (from 100 B to 10 KB) at high packet rate (250 packets per second). In the overlay network case it takes about 50 ms, with limited rise at increasing

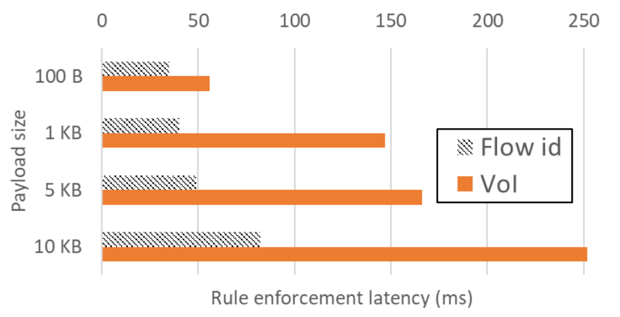

FIGURE 5. Rule enforcement cost at varying payload size (250 packets per second).

payload size, mainly due to slightly more complex memory management. In the VoI-aware case it takes from 50 ms for 100 B payload to more than 250 ms for 10 KB payload. In this case the payload size relevantly impacts on the achieved performance, due to the additional time required for deserialization and to inspect the payload content.

Overall, performance results achieved on a real-world testbed composed of Raspberry Pi devices based on our Java prototype demonstrate the feasibility of the proposed solution. In fact, by adopting the proposed SDN-based MLR solution it is possible to remotely deploy new service components in a prompt manner (also in case of bandwidth saturation thanks to our priority-based flow dispatching mechanism), while the time required to enforce rules is limited also in the challenging case it is required to deserialize huge packets.

B. SIMULATION OF REFERENCE SCENARIO

We also evaluated the HORNET framework in a simulated environment with the primary goal of testing the articulated reference scenario discussed in Section III, composed of several devices and different services. In particular, simulated experiments allow to present how the the dynamic deployment and activation of services composition, triggered by the time-varying requirements and resources of the target environment, influence the service-specific and overall VoI. In the following subsections we first introduce the adopted simulator and the target use-case; then, we detail run experiments and discuss about achieved results.

1) THE PHILEAS SIMULATOR AND THE TARGET USE-CASE

For the purpose of this experiment we adopted Phileas, a discrete event simulator that enables the reproducible evaluation of Fog applications in a realistic environment, which we developed and released as open source at: `https://github.com/DSG-UniFE/phileas` [46]. Phileas allows to simulate applications built on the AIV service model, tracking the VoI of each information during the whole processing, from the generation of a message to its consumption.

In fact, the VoI associated to each message depends on several factors, i.e., information type, service characteristics, end user interests, context, etc., and typically changes significantly throughout the message lifetime. In the simulated simulator, at the moment of its generation each raw data message is assigned an initial VoI attribute, while the VoI of IO messages generated by service components is calculated from the VoI of messages that were processed to generate that message according to service-specific policies. Moreover, each message has a VoI decay profile that models the loss of value as time passes and the information travels from its originating source, according to the configuration of the entity that generated the message.

Phileas also models communications, adopting a pragmatic approach oriented to allow the accurate evaluation of the VoI produced by Fog services running in the simulated scenarios. More specifically, Phileas models communication latency by sampling from a random variable with a long tail distribution, in accordance with several research studies [49]–[51]. In addition, it adopts a logarithmic propagation loss, which is suited for urban environments. We believe this approach represents a reasonable tradeoff between model complexity and accuracy, as it allows to account message losses and communication delays while dismissing lower-level and protocol specific aspects such as transmission rate, interference modeling, etc.

As depicted in Fig. 6, we chose to implement the reference scenario of Section III in the downtown area of Washington DC, USA. More specifically, we setup 7 different data sources (4 environmental stations and 3 traffic cameras) in an area between the National Mall, Capitol Hill, and Union Station. We assume that these data sources will generate raw data messages with random and exponentially distributed initial VoI values. The VoI of raw data messages will decay exponentially in both space and time as messages get disseminated. The time between the generation of subsequent messages, i.e., the inter-generation time, and the message size are also random and exponentially distributed.

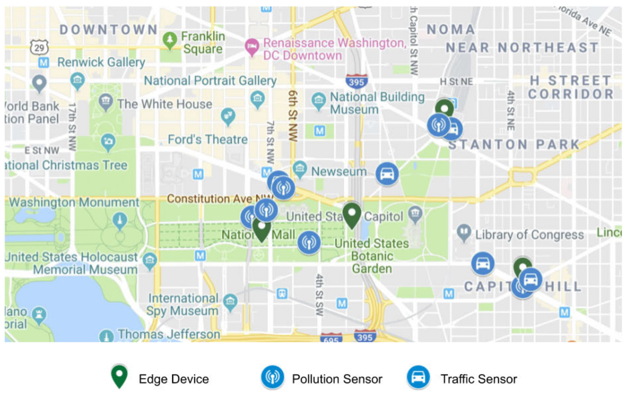

FIGURE 6. Positions of sensors and devices on the map as defined in the scenario.

TABLE 1. Characterization of service processing.

Name	$\Theta(m)$ distribution	$V_M^{(s)}$	Time decay type, half-life	Space decay type, half-life
TM	$DU(1,10)$	3.5	linear, 1000s	linear, 1km
PM	$DU(1,10)$	3.0	linear, 1000s	linear, 1km
PR	$DU(1,20)$	2.0	linear, 2000s	linear, 1km
FR	$DU(1,10)$	8.0	linear, 800s	linear, 1km

The raw data collected from IoT sensors serves as an input for the Fog services. At the beginning of the simulation, three services are running on the devices at the edge, analyzing data and disseminating the results in real-time: Traffic Monitoring (TM), Pollution Monitoring (PM), and Plate Recognition (PR). PM collects environmental data from local stations and generates reports with information about the air quality nearby. TM and PR gather video frames from traffic cameras in the downtown area and processes them, respectively, to assess the current viability status and to identify cars with plates of interest.

These services are implemented on top of 10 different components hosted in 4 devices deployed and managed by the municipality. Service components process incoming information according to a lossy and service-specific buffering policy: if they do not have enough resources they will drop messages when the buffer is full. In addition, to mimic the behavior of information processing services, we assume that an IO message m will be generated only after a random number $\Theta(m)$ of raw data messages, sampled from a probability distribution with service specific parameters, have been received. In the experiment, we adopted the discrete uniform distribution $DU(a, b)$ to model $\Theta(m)$.

For the purpose of these experiments, equation (1) is approximated as shown in eq. (5). The initial VoI VoI_0 of an IO message m is obtained by calculating the average VoI of raw data messages RD_i processed to obtain IO m and applying a service component specific multiplier $V_M^{(s)}$ as a weight that takes into account the overall parameters defined in equation (1). As with raw data messages, the VoI of IOs will decay in both space and time as they are disseminated, according to service specific policies and equations (2) and (3). For the purpose of these experiments, we considered linear time and space decay, with service specific half-life parameters. All the parameters we considered for service specific modeling are summarized in Table 1.

$$VoI_0(m) = V_M^{(s)} * \sum_{i=1}^{\Theta(m)} \frac{VoI(RD_i)}{\Theta(m)} \tag{5}$$

2) VoI-AWARE SERVICE MANAGEMENT

The objective of the simulations is to demonstrate that HORNET is able to promptly manage monitored Fog environments by appropriately considering a service components and network re-composition solution that optimizes the overall system VoI. To this purpose, we defined 6 user groups interested in the information provided by the services, with the characteristics described in Table 2. Groups 1-4 represent a mixture of citizens and municipality officers, while groups 5 and 6 represent police enforcement. Each user group modeled in the scenario has different share interests in receiving information from one of the three services, depending on the type of user and her/his location. While PM and TM provide valuable information for a wider range of users, PR is for municipality/police enforcement use only. We modeled the number of users at time t as a Gaussian random variable with distribution $N(\mu, \sigma)$.

We configured Phileas to reenact the scenario for 24 hours. We assume that after 4 hours from the beginning of the simulation a missing child report arrives, leading to the immediate activation of the Face Recognition (FR) service and of an additional device, as well as of another device one hour later. These changes to the set of service activated and resources available prompted HORNET to activate 5 additional service components (replicas) for FR between 4 and 7 hours from the beginning of the simulation to maximize the total VoI produced. In addition, after 6 hours from the beginning of the

TABLE 2. Characterization of user groups.

ID	Location	Subscriber share TM, PM, PR, FR	Size distribution
1	National Mall, in front of Art sculpture garden	65%, 30%, 30%, 45%	$N(150, 20)$
2	Madison Dr. and 7th St.	70%, 45%, 0%, 45%	$N(90, 20)$
3	Penn. Av. and D St.	60%, 85%, 0%, 35%	$N(115, 20)$
4	Close to Union Station	60%, 30%, 0%, 25%	$N(130, 20$
5	Penn. Av. and D St.	75%, 30%, 75%, 95%	$N(130, 20)$
6	National Mall, in front of Art sculpture garden	75%, 30%, 65%, 95%	$N(130, 20)$

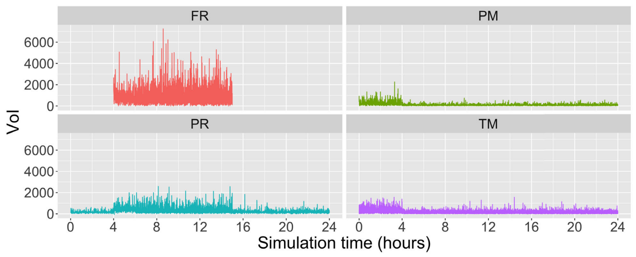

FIGURE 7. VoI produced by each service during the simulation.

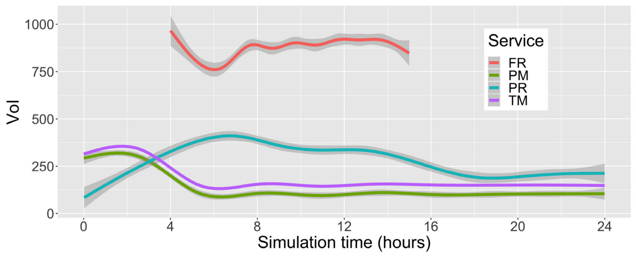

FIGURE 8. Comparison of (smoothed) VoI produced by each service during the simulation.

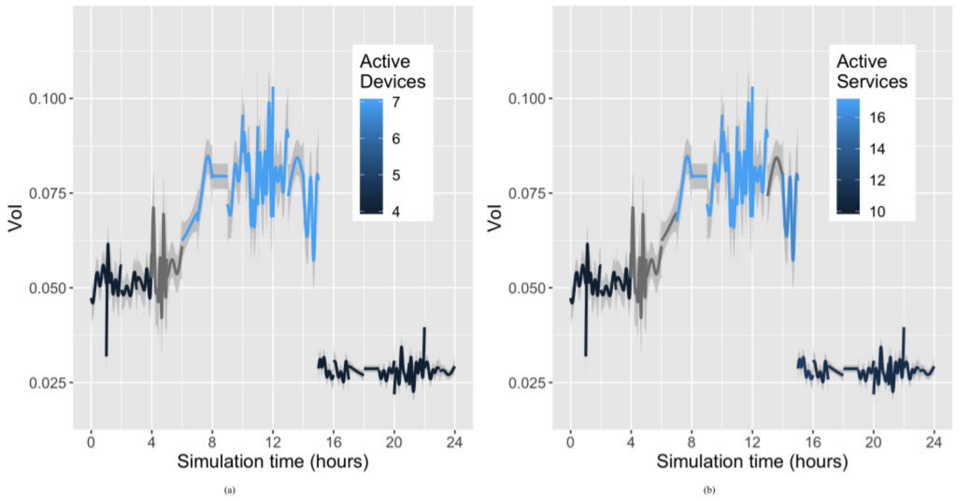

FIGURE 9. Total VoI (scaled) generated during the simulation, in correlation with: a) the number of active services (left), and b) the number of active devices (right).

simulation, a stolen car report arrives, leading to the increase of the VoI generated by PR (which we simulated by changing the corresponding value of $V_M^{(s)}$ to 6.0) and the activation of an additional device from the police. Again, the detection of a change in the VoI produced by PR, as well as of the availability of more resources, prompted HORNET to immediately allocate 2 additional service components (replicas) for PR. Then, to assess how HORNET would respond in case the stolen car and the missing case were found, we scheduled the deactivation of the 3 extra devices and reset the value of $V_M^{(s)}$ for PR between 14 and 17 hours from the beginning of the simulation.

As expected the VoI changes heavily throughout the simulation. Fig. 7 depicts with high granularity and at per service level the instantaneous VoI delivered by the IO delivered to users (y axis) versus the corresponding simulation time (x axis). Fig. 8 presents the same data but in a different form, comparing the curve VoI delivered by each service, smoothed through the interpolation at the entire simulation time level. Both figures clearly depict how the activation of new devices and of the face recognition service impacts the VoI provided to end users of Fog Computing applications. In fact, note how the activation of FR impacts TM and PM in a negative way, even after the activation of new devices. At the same time, the increased VoI of PR after the stolen car report event means that the service receives enough resources to deliver a good amount of VoI. Finally, after the deactivation of FR and of the on demand devices, the VoI curves return to the state at the beginning of the simulation.

Fig. 9 provides a different insight on the behaviour of HORNET. It shows the normalized and interpolated VoI accordingly to a color scale based on the number of active services (Fig. 9.a, on the left) and devices (Fig. 9.b, on the right). More specifically, we normalized the VoI value of each output messages in a [0, 1] range and we choose a 1-hour time interpolation window. The figure clearly shows that a high total

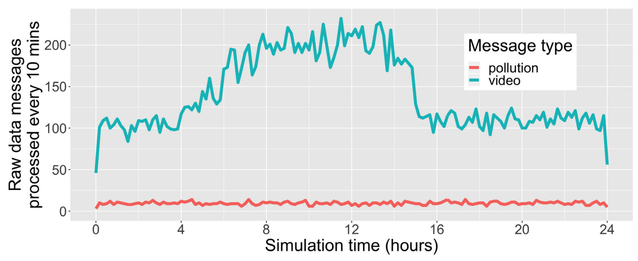

FIGURE 10. Number of raw data messages processed every 10 minutes throughout the simulation.

VoI is correlated with a high number of active services instantiated and/or devices available for computation, as this allows more valuable information to be disseminated to interested users. This is also confirmed by the data in Fig. 10, which shows the number of raw data messages during the simulation. The figure illustrates the number of raw data messages processed every 10 minutes during the simulation, divided per message type. These results demonstrate how the service composition stemming from a triggering-event (the need of locating a missing child) affects the VoI, thus highlighting the framework effectiveness in dynamically (re)allocating resources to optimize the total VoI generated at the system level.

Let us also note that the computational overhead introduced by the VoI estimation is negligible. In fact, the VoI approach implemented within HORNET allows to selectively filter raw-data messages, thus resulting in a considerably lower amount of information to be processed by service components. Therefore, the computational overhead introduced by the VoI estimation is dramatically lower than the computational resource-saving due to the VoI filtering. To quantify these values, let us specify that during the simulation time only 23.6% of the collected video frames and 14.4% of pollution samples were processed.

Overall, results achieved with the Phileas simulator highlight the effectiveness of the integrated VoI-based prioritization at both the service and communication layer performed by HORNET. In fact, by considering the time-varying VoI it is possible to dynamically deploy service components when and where needed, with the positive consequence of increasing the overall VoI itself. In other words, achieved results confirm the capability of HORNET to leverage the availability of more devices to improve the total VoI produced.

VI. CONCLUSION AND FUTURE WORK

The development of Fog Computing applications can significantly benefit from innovative solutions designed to prioritize the processing of the most valuable portion of information and to disseminate the results in a context-aware fashion. To this end, we devised the HORNET SDN-based solution, which enables the effective use of the scarce and heterogeneous computation and bandwidth resources in Fog Computing environments, by adopting an ''acceptable lossiness'' perspective together with the MLR approach. HORNET significantly facilitates the development of IoT applications while addressing the most important challenges that are raised by deployment environments such as the reference scenario discussed in Section III.

The experimental evaluation presented in Section V demonstrates the HORNET effectiveness in addressing optimal service component re-composition and MLR-based network configuration capable of maximizing the overall VoI in the simulated reference scenario. More specifically, results in Section V-A show that the HORNET Java prototype is capable of implementing fast (50-500ms) network layer reconfigurations, allowing quasi real-time responsiveness even in the worst case of large packets processed on resource limited hardware such as Raspberry Pi devices. In addition, the experiments in V-B demonstrate the effectiveness and robustness of VoI resource allocation in a simulated environment reenacting a highly dynamic scenario with multiple concurrent and heterogeneous services, thus validating the HORNET model and approach.

Encouraged by the already achieved and promising results reported in this paper, we are currently working on the HORNET middleware implementation to further increase its flexibility. To this purpose, our primary ongoing work at the moment is the extension of HORNET to support the dynamic and flexible federation of hierarchically organized SDN controllers, each one in charge of managing a part (i.e., a locality) of the Fog environment.

At a later stage, we intend to investigate the refinement of VoI evaluation models – a topic that scientific literature has relatively neglected so far. We believe that this represents a concern that goes beyond the aims and scope of the present proposal. However, in the future we aim at performing a step towards practical adoption of VoI based solutions by reducing or eliminating altogether the theoretical possibility that services competing for the same set of resources might receive unfair treatment.

REFERENCES

[1] J. Gedeon, F. Brandherm, R. Egert, T. Grube, and M. Muhlhauser, ''What the fog? Edge computing revisited: Promises, applications and future challenges,'' *IEEE Access*, vol. 7, pp. 152847–152878, 2019.

[2] M. Mukherjee, L. Shu, and D. Wang, ''Survey of fog computing: Fundamental, network applications, and research challenges,'' *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 1826–1857, 3rd Quart., 2018.

[3] M. Chiang and T. Zhang, ''Fog and IoT: An overview of research opportunities,'' *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[4] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, ''On reducing IoT service delay via fog offloading,'' *IEEE Internet Things J.*, vol. 5, no. 2, pp. 998–1010, Apr. 2018.

[5] M. De Donno, K. Tange, and N. Dragoni, ''Foundations and evolution of modern computing paradigms: Cloud, IoT, edge, and fog,'' *IEEE Access*, vol. 7, pp. 150936–150948, 2019.

[6] J. Santos, T. Wauters, B. Volckaert, and F. De Turck, ''Fog computing: Enabling the management and orchestration of smart city applications in 5G networks,'' *Entropy*, vol. 20, no. 1, p. 4, 2018.

[7] L. Li, Q. Guan, L. Jin, and M. Guo, ''Resource allocation and task offloading for heterogeneous real-time tasks with uncertain duration time in a fog queueing system,'' *IEEE Access*, vol. 7, pp. 9912–9925, 2019.

[8] L. Zhang and J. Li, ''Enabling robust and privacy-preserving resource allocation in fog computing,'' *IEEE Access*, vol. 6, pp. 50384–50393, 2018.

[9] A. Brogi and S. Forti, ''QoS-aware deployment of IoT applications through the fog,'' *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1185–1192, Oct. 2017.

[10] Y. Yin, W. Zhang, Y. Xu, H. Zhang, Z. Mai, and L. Yu, ''QoS prediction for mobile edge service recommendation with auto-encoder,'' *IEEE Access*, vol. 7, pp. 62312–62324, 2019.

[11] X. Zhang, S. Yu, J. Zhang, and Z. Xu, ''Forwarding rule multiplexing for scalable SDN-based Internet of things,'' *IEEE Internet Things J.*, vol. 6, no. 2, pp. 3373–3385, Apr. 2019.

[12] P. K. Sharma, M.-Y. Chen, and J. H. Park, ''A software defined fog node based distributed blockchain cloud architecture for IoT,'' *IEEE Access*, vol. 6, pp. 115–124, 2018.

[13] H. Xiang, W. Zhou, M. Daneshmand, and M. Peng, ''Network slicing in fog radio access networks: Issues and challenges,'' *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 110–116, Dec. 2017.

[14] N. Suri, G. Benincasa, R. Lenzi, M. Tortonesi, C. Stefanelli, and L. Sadler, ''Exploring value-of-information-based approaches to support effective communications in tactical networks,'' *IEEE Commun. Mag.*, vol. 53, no. 10, pp. 39–45, Oct. 2015.

[15] F. Poltronieri, M. Tortonesi, A. Morelli, C. Stefanelli, and N. Suri, ''Value of information based optimal service fabric management for fog computing,'' in *Proc. IEEE-IFIP Netw. Oper. Manage. Symp. (NOMS)*, Budapest, Hungary Apr. 2020, pp. 20–24.

[16] C. Giannelli, F. Poltronieri, C. Stefanelli, and M. Tortonesi, ''Supporting the development of next-generation fog services,'' in *Proc. IEEE 23rd Int. Workshop Comput. Aided Model. Design Commun. Links Netw. (CAMAD)*, Sep. 2018, pp. 1–6.

[17] R. Jain and S. Paul, ''Network virtualization and software defined networking for cloud computing: A survey,'' *IEEE Commun. Mag.*, vol. 51, no. 11, pp. 24–31, Nov. 2013.

[18] Y. Bi, G. Han, C. Lin, Q. Deng, L. Guo, and F. Li, ''Mobility support for fog computing: An SDN approach,'' *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 53–59, May 2018.

[19] A. C. Baktir, A. Ozgovde, and C. Ersoy, ''How can edge computing benefit from software-defined networking: A survey, use cases, and future directions,'' *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2359–2391, 4th Quart., 2017.

[20] E. Datsika, A. Antonopoulos, N. Zorba, and C. Verikoukis, ''Software defined network service chaining for OTT service providers in 5G networks,'' *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 124–131, Nov. 2017.

[21] R. Vilalta, V. Lopez, A. Giorgetti, S. Peng, V. Orsini, L. Velasco, R. Serral-Gracia, D. Morris, S. De Fina, F. Cugini, P. Castoldi, A. Mayoral, R. Casellas, R. Martinez, C. Verikoukis, and R. Munoz, ''TelcoFog: A unified flexible fog and cloud computing architecture for 5G networks,'' *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 36–43, Aug. 2017.

[22] F. Y. Okay and S. Ozdemir, ''Routing in fog-enabled IoT platforms: A survey and an SDN-based solution,'' *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4871–4889, Dec. 2018.

[23] C. Li, Z. Qin, E. Novak, and Q. Li, ''Securing SDN infrastructure of IoT–Fog networks from MitM attacks,'' *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1156–1164, Oct. 2017.

[24] Q. Shafi, A. Basit, S. Qaisar, A. Koay, and I. Welch, ''Fog-assisted SDN controlled framework for enduring anomaly detection in an IoT network,'' *IEEE Access*, vol. 6, pp. 73713–73723, 2018.
[25] W. Villota, M. Gironza, A. Ordonez, and O. M. C. Rendon, ''On the feasibility of using hierarchical task networks and network functions virtualization for managing software-defined networks,'' *IEEE Access*, vol. 6, pp. 38026–38040, 2018.
[26] Y. Li and M. Chen, ''Software-defined network function virtualization: A survey,'' *IEEE Access*, vol. 3, pp. 2542–2553, 2015.
[27] R. Yu, G. Xue, and X. Zhang, ''QoS-aware and reliable traffic steering for service function chaining in mobile networks,'' *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2522–2531, Nov. 2017.
[28] H. Hawilo, M. Jammal, and A. Shami, ''Network function virtualization-aware orchestrator for service function chaining placement in the cloud,'' *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 643–655, Mar. 2019.
[29] D. Zhao, J. Ren, R. Lin, S. Xu, and V. Chang, ''On orchestrating service function chains in 5G mobile network,'' *IEEE Access*, vol. 7, pp. 39402–39416, 2019.
[30] L. M. Vaquero, F. Cuadrado, Y. Elkhatib, J. Bernal-Bernabe, S. N. Srirama, and M. F. Zhani, ''Research challenges in nextgen service orchestration,'' *Future Gener. Comput. Syst.*, vol. 90, pp. 20–38, Jan. 2019.
[31] H. Gao, W. Huang, Y. Duan, X. Yang, and Q. Zou, ''Research on cost-driven services composition in an uncertain environment,'' *J. Internet Technol.*, vol. 20, no. 3, pp. 755–769, 2019.
[32] H. Gao, Y. Duan, L. Shao, and X. Sun, ''Transformation-based processing of typed resources for multimedia sources in the IoT environment,'' *Wireless Netw.*, pp. 1–17, Nov. 2019. [Online]. Available: https://link.springer.com/article/10.1007/s11276-019-02200-6
[33] Y. Chen, S. Deng, H. Ma, and J. Yin, ''Deploying data-intensive applications with multiple services components on edge,'' *Mobile Netw. Appl.*, pp. 1–16, Apr. 2019. [Online]. Available: https://link.springer.com/article/10.1007/s11036-019-01245-3
[34] C. Zhang, H. Zhao, and S. Deng, ''A density-based offloading strategy for IoT devices in edge computing systems,'' *IEEE Access*, vol. 6, pp. 73520–73530, 2018.
[35] H. Gao, Y. Xu, Y. Yin, W. Zhang, R. Li, and X. Wang, ''Context-aware qos prediction with neural collaborative filtering for internet-of-Things services,'' *IEEE Internet Things J.*, early access, Dec. 2, 2019, doi: 10.1109/JIOT.2019.2956827.
[36] C. Bisdikian, L. M. Kaplan, and M. B. Srivastava, ''On the quality and value of information in sensor networks,'' *ACM Trans. Sensor Netw.*, vol. 9, no. 4, pp. 1–26, Jul. 2013.
[37] D. Turgut and L. Boloni, ''Value of information and cost of privacy in the Internet of Things,'' *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 62–66, Sep. 2017.
[38] S. Bharti, K. K. Pattanaik, and P. Bellavista, ''Value of information based sensor ranking for efficient sensor service allocation in service oriented wireless sensor networks,'' *IEEE Trans. Emerg. Topics Comput.*, early access, Jan. 9, 2019, doi: 10.1109/TETC.2019.2891716.
[39] G. Xu, E. C.-H. Ngai, and J. Liu, ''Ubiquitous transmission of multimedia sensor data in Internet of Things,'' *IEEE Internet Things J.*, vol. 5, no. 1, pp. 403–414, Feb. 2018.
[40] P. Bellavista, A. Corradi, and C. Giannelli, ''Middleware for differentiated quality in spontaneous networks,'' *IEEE Pervas. Comput.*, vol. 11, no. 3, pp. 64–75, Mar. 2012.
[41] M. Tortonesi, M. Govoni, A. Morelli, G. Riberto, C. Stefanelli, and N. Suri, ''Taming the IoT data deluge: An innovative information-centric service model for fog computing applications,'' *Future Gener. Comput. Syst.*, vol. 93, pp. 888–902, Apr. 2019.
[42] C. Giannelli, P. Bellavista, and D. Scotece, ''Software defined networking for quality-aware management of multi-hop spontaneous networks,'' in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Maui, HI, USA, Mar. 2018, pp. 561–566.
[43] P. Bellavista, A. Dolci, and C. Giannelli, ''MANET-oriented SDN: Motivations, challenges, and a solution prototype,'' in *Proc. IEEE 19th Int. Symp. World Wireless, Mobile Multimedia Networks (WoWMoM)*, Chania, Greece, Jun. 2018, pp. 14–22.
[44] J. Yu, M. Tan, H. Zhang, D. Tao, and Y. Rui, ''Hierarchical deep click feature prediction for fine-grained image recognition,'' *IEEE Trans. Pattern Anal. Machine Intell.*, early access, Jul. 30, 2019, doi: 10.1109/TPAMI.2019.2932058.
[45] J. Yu, J. Li, Z. Yu, and Q. Huang, ''Multimodal transformer with multi-view visual representation for image captioning,'' *IEEE Trans. Circuits Syst. Video Technol.*, early access, Oct. 15, 2019, doi: 10.1109/TCSVT.2019.2947482.
[46] F. Poltronieri, C. Stefanelli, N. Suri, and M. Tortonesi, ''Phileas: A simulation-based approach for the evaluation of value-based fog services,'' in *Proc. IEEE 23rd Int. Workshop Comput. Aided Model. Design Commun. Links Netw. (CAMAD)*, Sep. 2018, pp. 1–6.
[47] S. Marsili Libelli and P. Alba, ''Adaptive mutation in genetic algorithms,'' *Soft Comput.*, vol. 4, no. 2, pp. 76–80, Jul. 2000.
[48] H. Cobb, ''An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments,'' Naval Research Lab, Washington DC, USA, Tech. Rep. AIC-90-001, 1990.
[49] C. Pei, Y. Zhao, G. Chen, R. Tang, Y. Meng, M. Ma, K. Ling, and D. Pei, ''WiFi can be the weakest link of round trip network latency in the wild,'' in *Proc. IEEE INFOCOM - 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.
[50] K. Sui, M. Zhou, D. Liu, M. Ma, D. Pei, Y. Zhao, Z. Li, and T. Moscibroda, ''Characterizing and improving WiFi latency in large-scale operational networks,'' in *Proc. 14th Annu. Int. Conf. Mobile Syst., Appl., Services MobiSys*, 2016, pp. 347–360.
[51] T. Høiland-Jørgensen, P. Hurtig, and A. Brunstrom, ''The good, the bad and the WiFi: Modern AQMs in a residential setting,'' *Comput. Netw.*, vol. 89, pp. 90–106, Oct. 2015.

PAOLO BELLAVISTA (Senior Member, IEEE) received the Ph.D. degree in computer engineering from the University of Bologna, Italy, in 2001. He is currently a Full Professor of distributed and mobile systems at the CSE Department, University of Bologna. His primary research activities span from mobile middleware to wireless sensor and actuator networks, from pervasive mobile computing infrastructures to the industrial Internet of Things, from edge cloud computing to online stream processing in manufacturing industry applications.

CARLO GIANNELLI (Member, IEEE) received the Ph.D. degree in computer engineering from the University of Bologna, Italy, in 2008. He is currently an Associate Professor in computer science with the University of Ferrara, Italy. His primary research activities focus on the industrial Internet of Things, software-defined networking, blockchain technologies, location-based services, heterogeneous wireless interface integration, and hybrid infrastructure/ad hoc and spontaneous multihop networking environments based on social relationships.

DMITRIJ DAVID PADALINO MONTENERO (Student Member, IEEE) received the master's degree in computer engineering from the Interdepartmental Centres for Industrial Research, University of Bologna, Italy, in 2019. He is currently an Industrial Research Fellow with the Interdepartmental Centres for Industrial Research, University of Bologna. His primary research activities focus on cloud computing, containerization, industrial Internet of Things, software-defined networking, and hybrid infrastructure/ad hoc and spontaneous multihop networking environments.

FILIPPO POLTRONIERI (Student Member, IEEE) is currently pursuing the Ph.D. degree with the Engineering Department, University of Ferrara, Ferrara, Italy. He joined the Distributed System Research Group, led by Prof. Cesare Stefanelli, in 2017. He visited the Florida Institute for Human and Machine Cognition (IHMC), Pensacola, FL, USA, from 2016 to 2018. His research interests include distributed systems, the IoT, Fog Computing, and tactical networks.

CESARE STEFANELLI (Member, IEEE) received the Ph.D. degree in computer science engineering from the University of Bologna, Italy, in 1996. He is currently a Full Professor of distributed systems with the Engineering Department, University of Ferrara, Italy. At the University of Ferrara he coordinates a Technopole Laboratory dealing with industrial research and technology transfer. He holds several patents, and coordinates industrial research projects carried on in collaboration with several companies. His research interests include distributed and mobile computing in wireless and ad hoc networks, network and systems management, and network security.

MAURO TORTONESI (Member, IEEE) received the Ph.D. degree in computer engineering from the University of Ferrara, Italy, in 2006. He was a Visiting Scientist with the Florida Institute for Human and Machine Cognition (IHMC), Pensacola, FL, USA, from 2004 to 2005 and with the United States Army Research Laboratory, Adelphi, MD, USA, in 2015. He is currently an Associate Professor with the Department of Mathematics and Computer Science, University of Ferrara. He holds two international patents and participates to the editorial board of four international scholarly journals. He has coauthored over 80 articles published in international venues in the distributed systems research area, with particular reference to the IoT solutions in industrial and military environments, cloud and fog computing, wireless middleware, and IT service management.