Mobile Edge Cloud System: Architectures, Challenges, and Approaches

Hang Liu, *Senior Member, IEEE*, Fahima Eldarrat, Hanen Alqahtani, Alex Reznik, *Senior Member, IEEE*, Xavier de Foy, and Yanyong Zhang, *Senior Member, IEEE*

***Abstract*—Mobile edge cloud (MEC) is a model for enabling on-demand elastic access to, or an interaction with a shared pool of reconfigurable computing resources such as servers, storage, peer devices, applications, and services, at the edge of the wireless network in close proximity to mobile users. It overcomes some obstacles of traditional central clouds by offering wireless network information and local context awareness as well as low latency and bandwidth conservation. This paper presents a comprehensive survey of MEC systems, including the concept, architectures, and technical enablers. First, the MEC applications are explored and classified based on different criteria, the service models and deployment scenarios are reviewed and categorized, and the factors influencing the MEC system design are discussed. Then, the architectures and designs of MEC systems are surveyed, and the technical issues, existing solutions, and approaches are presented. The open challenges and future research directions of MEC are further discussed.**

***Index Terms*—Fog computing, mobile edge cloud (MEC), mobile edge computing, service model, survey, system architecture.**

I. Introduction

THE trend toward the integration of advanced information technologies (IT) and communications technologies is creating unprecedented opportunities in the design of innovative network infrastructure to improve user experience and enable a broad range of new internet applications. Rapid growth in cloud computing is an example of such integration. However, the traditional cloud computing model that depends on centralized data centers faces several severe limitations, especially for emerging mobile and Internet of Things (IoT) applications [1], [2]. First, it is costly or undesirable to move all the data collected at the edge, e.g., all the video surveillance data in a metropolitan area or other data generated by IoT applications, to the central cloud over the Internet for real-time processing. Second, for time-critical applications, sending data to distant data centers over the Internet may not always be able to meet their latency requirements. For example, some mobile games require fast responses of less than 100 ms and some tactile Internet applications such as interactive speech translation requires response time less than 10 ms. Third, for applications that need adaptation to the local network status and local user context, e.g., streaming a live video to a mobile user, it is difficult to efficiently adapt traffic (video coding rate) to respond to quick changes on available wireless bandwidth using far-away data centers.

To address these limitations, a new model that integrates computing and storage capabilities at the edge of the network, closer to the rapidly growing number of mobile devices, is recently gaining attention in both academia and industry. Several similar concepts with different terminologies have been proposed. Edge clouds or mobile edge clouds (MECs) equip or colocate computation and storage components with edge networking elements such as base stations (BSs), access points (APs), switches, routers, and/or pool the computation and storage resources of mobile devices over an edge network to perform distributed compute tasks [3]–[10]. Fog computing [11]–[13] is a virtualized platform concept, initiated by Cisco and several other major industry players, which is typically located at the edge of networks to provide compute, storage, and networking services between mobile end devices and traditional data centers. It can also be viewed more generally as a scenario in which a large number of heterogeneous devices communicate and potentially cooperate among them and with the network to perform storage and processing tasks, and mobile users may lease part of their devices to host these services and get incentives for doing so [14]–[17]. Mobile edge computing [18]–[21], another similar concept, provides IT and cloud-computing capabilities within radio access networks (RANs) at the close proximity to mobile subscribers. The mobile edge computing server may be deployed at BS, radio network controller, and a site with a number of multitechnology APs. The initial industry standardization effort on mobile edge computing has started in the European Telecommunications Standards Institute (ETSI) [22] with a focus on managed mobile cellular networks to unite the telco and IT-cloud worlds, providing IT and cloud-computing capabilities within the RANs.

In this paper, we use the term "mobile edge cloud" along with its acronym "MEC." We define MEC as a model for enabling on-demand elastic access to, or an interaction with a shared pool of reconfigurable computing resources such as servers, storage, peer devices, applications, and services, with minimal management effort and service provider intervention in wireless access

Manuscript received June 1, 2016; revised September 9, 2016 and November 15, 2016; accepted January 7, 2017. Date of publication February 6, 2017; date of current version August 23, 2018.

H. Liu, F. Eldarrat, and H. Alqahtani are with the Electrical Engineering and Computer Science Department, Catholic University of America, Washington, DC 20064 USA (e-mail: liuh@cua.edu; 40eldarrat@cua.edu; 94alqahtani@cua.edu).

A. Reznik was with InterDigital Communications, Inc., Conshohocken, PA 19428 USA. He is now with the Hewlett Packard Enterprise Company, New York, NY 10011 USA (e-mail: rezlev@gmail.com).

X. de Foy is with InterDigital Communications, Inc., Montreal, QC H3A 3G4, Canada (e-mail: xavier.defoy@interdigital.com).

Y. Zhang is with WINLAB, Rutgers University, North Brunswick, NJ 08902 USA (e-mail: yyzhang@ece.rutgers.edu).

Digital Object Identifier 10.1109/JSYST.2017.2654119

networks at close proximity to mobile users. MEC distinguishes from traditional cloud computing that depends on centralized data centers, and mobile cloud computing that combines cloud computing, mobile computing, and wireless networks and allows the cloud to handle large storage and processing for mobile devices [23]–[28].

The research on MEC is still in its infancy. Several position papers and survey papers [12], [14], [15], [29], [30] mainly focus on the MEC concept, application scenarios, characteristics, general issues, and high-level directions. This paper provides a comprehensive survey of the architectures and designs of MEC systems from a technical point of view, and outlines the existing solutions of several key technical issues. The remainder of this paper is organized as follows. Section II summarizes the characteristics of MEC, and classifies the MEC applications based on different criteria. The service models and deployment scenarios are categorized, and the factors influencing MEC system design are discussed in Section III. In Section IV, the architectures and designs of MEC systems are surveyed, and the technical issues, existing solutions, and approaches are presented. The open challenges and future research directions of MEC are discussed in Section V. Section VI concludes this paper.

II. MEC Characteristics and Applications

Fig. 1 shows an MEC system, in which MEC servers are deployed in a wireless access network and provide computing and storage capabilities. By offering computation and storage capabilities at the edge of the network, MEC can minimize latency, conserve network bandwidth, reduce cross-domain traffic, make better location- and context-aware decisions, as well as alleviate security and privacy concerns. It will allow network operators to provide additional value-added services and improve the quality of experience (QoE) of end users by leveraging the unique characteristics of their networks. In addition, network operators may authorize third-parties to access their infrastructure. Application developers and content providers can utilize MEC platform to offer context-aware services to mobile users in close proximity. This ecosystem could facilitate innovation for new applications that would not be possible otherwise or not well served by traditional distant clouds. MEC has the following characteristics [12], [18].

1) Local network status awareness: MEC is deployed at the edge, and is able to access real-time wireless network and channel information.
2) Local user context awareness: applications can leverage location and user context locally.
3) Distributed: MEC resources, applications, and services may geographically distribute at different locations and hierarchy levels of a wireless access network, such as BSs, APs, switches, routers, gateways, as well as mobile devices.
4) Heterogeneity: in contrast to dedicated data centers with well-planned devices and networks, MEC nodes are heterogeneous with varying processing and storage resources as well as varying network connectivity and bandwidth.
5) Mobility and unreliable access: mobile devices access MEC services through unreliable wireless access, and often change their points of attachment to the network. MEC services themselves may be hosted at mobile devices. Mobility support is critical for MEC.
6) Ultra-low latency: MEC should potentially achieve extremely low latency, e.g., one or several milliseconds, to support tactile Internet applications such as robotics, virtual and augmented reality (AR), and real-time interactive industrial control systems. These applications are highly sensitive to delay and require for the end-to-end latency on the order of human tactile response time.
7) Interplay with central cloud: MECs are complementary to traditional central clouds, not to replace them. The MECs are close to mobile devices and are able to provide localized processing with context awareness and low latency, while the distant traditional clouds offer global centralization with much more powerful computing and storage capacity. The execution of many applications may need to span across mobile end devices, MEC network nodes, and distant clouds.

The features of MEC ensure a wide range of the applications that can benefit from being deployed at the edge. There are different methods to abstract and categorize the MEC applications as shown in Fig. 2. Based on the properties, the MEC applications include the following.

1) Latency-sensitive applications. As the process can be executed at the edge and data do not have to travel over the Internet back and forth between end devices and remote data centers, the latency will be reduced and the user experience of delay sensitive applications such as real-time traffic control and networked gaming will be improved.
2) Applications that need to adapt to the local network status and requirements. For example, as the radio channel condition deteriorates or network traffic load increases, edge applications such as in-network video transcoding can adapt the video data rate to meet the reduction of available bandwidth. In addition, network operators can utilize edge computing capability to deploy firewalling, VPN, and other in-network services based on the local network requirements.
3) Applications with local nature or local context. Local applications such as local content sharing, insertion of local ads, local crowdsensing, etc., can leverage location information, user context, and local computing capability to improve performance because the data processing and computation are conducted on smaller datasets locally with less overhead and latency. It also increases privacy as the location of mobile users is not sent out of the wireless access network.
4) Applications that need to process and aggregate data. As a lot of data are collected at the edge of the network for certain applications, such as video surveillance, security monitoring, and big data analysis, MEC can be used to process, analyze, and aggregate data, the results are then sent to the central cloud. The processing at the edge will significantly reduce the amount of data that need to be shipped to the central cloud over the Internet, and conserve the bandwidth and cloud storage.

Furthermore, there may be much higher bandwidth in wireless access networks than on the cross-domain Internet links. An application program may consist of multiple tasks. Here, a task

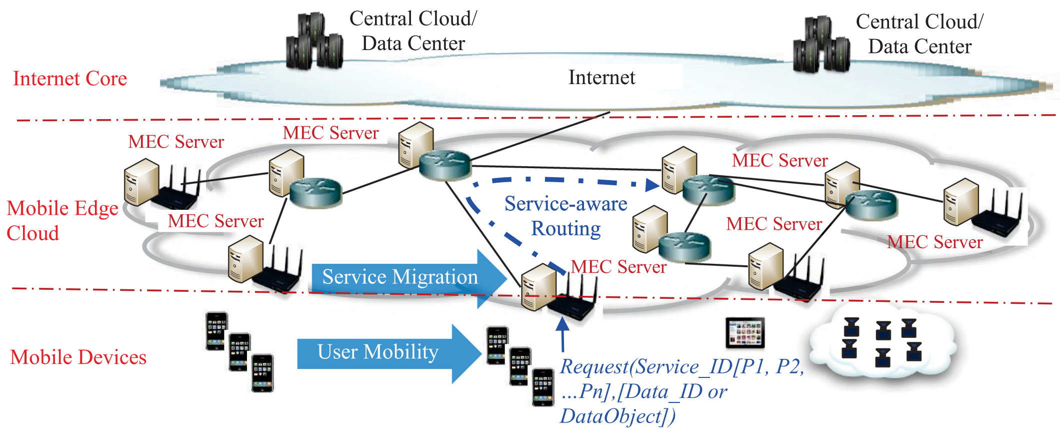

Fig. 1. Mobile edge cloud system overview.

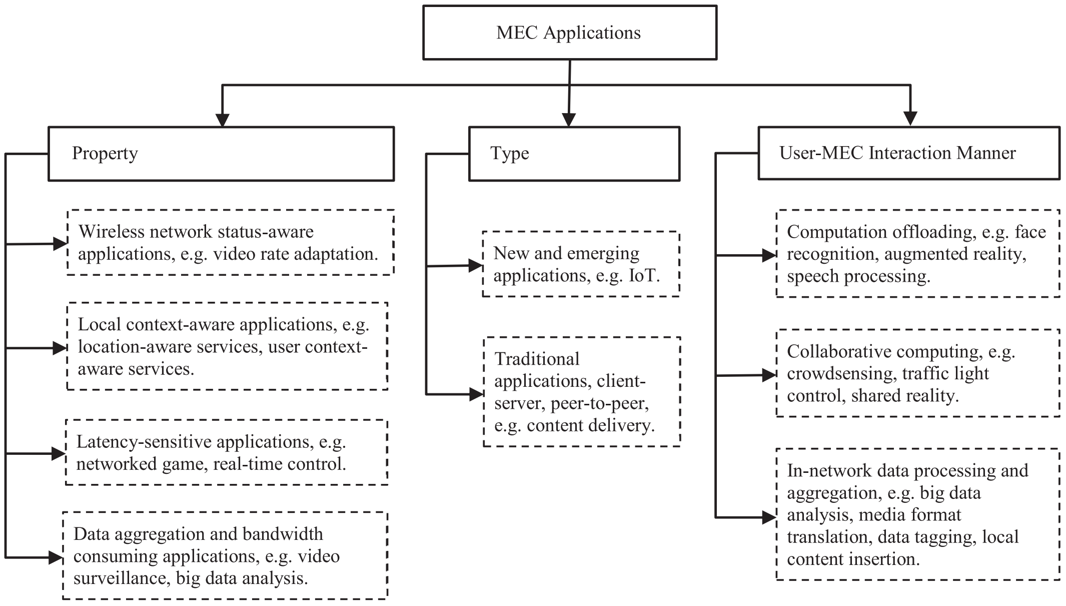
MEC Applications
Property
Type
User-MEC Interaction Manner
Wireless network status-aware applications, e.g. video rate adaptation.
Local context-aware applications, e.g. location-aware services, user context-aware services.
Latency-sensitive applications, e.g. networked game, real-time control.
Data aggregation and bandwidth consuming applications, e.g. video surveillance, big data analysis.
New and emerging applications, e.g. IoT.
Traditional applications, client-server, peer-to-peer, e.g. content delivery.
Computation offloading, e.g. face recognition, augmented reality, speech processing.
Collaborative computing, e.g. crowdsensing, traffic light control, shared reality.
In-network data processing and aggregation, e.g. big data analysis, media format translation, data tagging, local content insertion.

Fig. 2. MEC application classifications.

is a piece of computation work or a function to be performed. These tasks can be executed with the processes or threads distributed on different physical or virtual machines (VMs). Some complex tasks could be offloaded from the mobile devices to the more powerful edge nodes with minimal latency. MEC will thus make feasible certain time-critical and high-bandwidth applications that cannot be served well before. We notice that many applications possess more than one of the above-mentioned characteristics.

Alternatively, the MEC applications can be classified as emerging IoT applications and conventional applications. IoT applications involve sensing, event detection, control of actuators, as well as interactions among machines or between human and machines. Many IoT applications not only consume a lot of data but also generate a lot of traffic at the edge of network, and often incur a sense-process-actuate loop. Conventional applications more involve server–client interaction and peer-to-peer traffic. IoT applications are one of the main drivers of MEC. First, hundreds of billions of devices consuming and producing data at the edge of the network lead to demand of processing and aggregation at the network edge, thus only aggregates may be sent or published over the Internet to alleviate bandwidth needs. Second, real-time machine-to-machine monitoring, analysis, and control loops require MEC for reduced latency. Third, low resource and battery-operated sensors and actuators need to offload the computation and storage to more powerful devices.

According to the manner of user–MEC interaction, the applications can be classified as computation offloading, collaborative computing, and in-network processing. Computation offloading allows a resource-constrained mobile device to offload computation-intense tasks to MEC servers or other MEC devices, and the results are returned after successfully executing the tasks. Mobile devices may not be able to execute all the tasks in time because their computation capabilities are limited or the computation will cause very high power consumption and deplete their battery power quickly [29], [30]. For example, compute-intensive applications such as AR, face recognition, and speech processing, can be split into multiple tasks and some of the tasks are performed at the MEC servers. Offloading increases the capability of mobile devices and reduces their power consumption. Although mobile applications may offload computation tasks to the central cloud, offloading at the MEC without transferring the tasks over the Internet will certainly reduce latency, save cross-domain bandwidth usage, provide better privacy, and other benefits as discussed above.

Collaborative computing enables multiple devices in a MEC to collaborate each other to achieve a job. It provides a real-time context-aware distributed computing system within the edge network. For example, in the smart traffic control scenario [31], the cameras and sensor devices installed along the roadside detect the vehicles and pedestrian, and measure their speed and distance. The real-time data collected are sent to a traffic controller running at an MEC node that analyzes the data and control multiple traffic lights to direct vehicles and pedestrian. Similarly, in the vehicle identification and tracking applications, traffic surveillance cameras can help police identify and track vehicles for which they have issued a search. Smart cameras in the area detect motion in the scene and send video to the recognition service deployed at the MEC nodes that run computer vision algorithms to identify the suspect's vehicle and control the cameras pan-tilt-zoom to get clear view and track the vehicle. Once a match is found, the police will be notified and receive the video tracking the suspect's vehicle.

In-network data processing allows processing, aggregating, and storing data in the MEC for performance optimization and improved user experience based on dynamic local network status and context as the data are sent from mobile devices to Internet data centers, from data centers to mobile devices, or from mobile devices to mobile devices. For example, many sensors collect related data and send it to the MEC for further processing, aggregation, and caching so as to provide some IoT services [14], [32]. The other example is that user-generated video or images will be downscaled at the MEC nodes before they are forwarded to the data centers over the Internet, which will significantly reduce Internet traffic and storage demands, and improve scalability of data centers. Another example is that if congestion occurs while transmitting multiple video streams to mobile users over a radio channel, MEC may transcode video to lower bitrates and optimize the overall user experience of multiple streams before sending them to the mobile users. In addition, MEC may examine the data for security based on the local network policy, transform media format according to the user device capability, or insert local content based on the user context before delivering data to users. It may also add network information such as available bandwidth before forwarding user requests to the remote application servers so the applications may utilize the information to better serve users [29].

III. MEC Service Models

Similar to cloud computing, MEC can provide three levels of services, software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS) to its customers. However, there are significant differences between MEC and cloud computing services as summarized in Table I and explained below.

TABLE I
Characteristics of MEC and Traditional Cloud Computing

	MEC	Traditional Cloud
Resource & service location	At different levels of the wireless access network, BSs, APs, switches, routers, gateways, and mobile devices (close to mobile users)	Dedicated data centers on Internet (data needs to be sent to the data centers over Internet)
Service context awareness	Aware of local radio network status and user context	Location and user context available via application reporting
Latency	Low latency to support tactile Internet applications (several milliseconds)	Higher latency
Resource capability	Relatively limited (micro-datacenters)	Much more powerful
Resource heterogeneity	More heterogeneous and distributed with varying processing and storage resources and network connectivity.	Well-planned devices and networks in machine rooms.
Mobility	Mobile clients, services hosed on mobile devices	Mobile and fixed clients, services hosted on fixed servers
Services	User requested and network initiated (transparent) services	User requested services

MEC infrastructure operators or owners control mobile access networks with built-in computation and storage resources, instead of large data centers connected to the Internet. They may deploy their own applications and services. They can also open their wireless access networks to third-party partners, such as application service providers, allowing them to deploy services to mobile users [18]. For MEC SaaS, mobile users simply use the applications without concerning system configuration problems, such as employing cloud computing applications in the data centers. However, these MEC application services will benefit from being in close proximity to the users along with low latency and from receiving local wireless network contextual information, as well as saving backhaul bandwidth. For example, in AR, the application on the MEC server can provide local object tracking and local AR content catching. This will minimize round trip time and maximizes throughput for the best QoE. In video analysis service, the video management application hosted in MEC transcodes and stores captured video streams from wireless cameras. The video analytics application processes the video data to detect and notify specific events, e.g., object movement, lost child, abandoned luggage, etc. The application sends low bandwidth video metadata to the central operations and management server for database searches. In addition, by caching data at the wireless network edge, it will reduce the download time and provide backhaul transport savings.

MEC application services could be user requested services and transparent services. The latter is activated by the service provider when the users' data pass through the mobile access networks. For example, network operators may deploy firewalls, intrusion detection and prevention systems, and WAN optimizers. Content providers may deploy transcoding or caching applications on MEC servers. These services are invoked

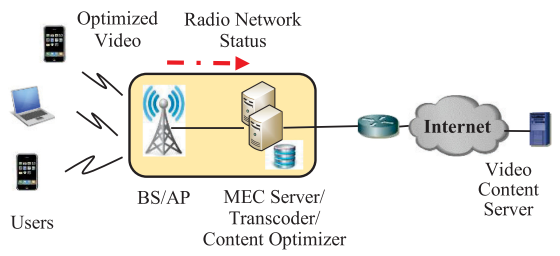

Fig. 3. Radio-aware content optimization.

automatically. In the radio-aware video content optimization as shown in Fig. 3, video streams are automatically transcoded to lower data rates in order to avoid stalling and improve users' QoE when the wireless network experiences congestion. However, traditional cloud computing services are normally initiated by the user applications.

To support MEC application services, no matter what are provided by an MEC operator or third party, there are a set of technical factors influencing the system design. They include how to name and address the services, how to deploy the services at appropriate points in the network, how to discover the services, how to route the requests to the best service instances in the network and invoke the services, how to seamlessly handle user mobility and service migration, how to slice and virtualize the network for different services, how to orchestrate and manage the services, how to rapidly provision and release resources as needed without requiring human interaction, etc.

On the other hand, MEC operators may provide their customers with a platform or an execution environment and middleware. The customers can create or deploy applications on the platform using tools supported by the provider. MEC platform services may have several use cases.

1) Third-party application service providers may lease a share of the platform, and deploy applications to serve mobile end users.
2) Mobile end users may deploy their own applications on the MEC platform, e.g., in crowdsensing.
3) Mobile end users may send a piece of program or droplet to the MEC platform service to execute the code automatically on the MEC platform.

However, the programing models and interfaces as well as available platform services for MEC may be different from those in cloud computing PaaS, for example, MEC platform will provide the radio network information and the interfaces to obtain the information to the applications. In the radio-aware content optimization, accurate radio network information such as cell load and link quality are exposed to the content optimizer, enabling dynamic content optimization to improve QoE and network efficiency. By tracking the location and context of active mobile users in the network and exposing the information to the MEC applications, it will enable location- and context-based services such as mobile advertising, traffic control, and smart city.

Mobile edge network operators may also offer IaaS to users. Users can provision processing, storage, and networking resources to build highly adaptable distributed systems, where they are able to deploy and run their software, which may include operating systems and applications. The MEC infrastructure services are mainly used for the development and deployment of platform and software services and applications to their end users by the application service providers or enterprise customers. Compared to the traditional cloud infrastructure, MEC resources are located at different hierarchy levels of the wireless access network without dedicated well-planed high-speed connections (even hosted on mobile devices), and along with varying processing and storage capabilities. They are much more distributed and heterogeneous than cloud data centers. In addition, mobility and unreliable wireless access should be carefully handled. This makes resource management and orchestration much more challenging.

Mobile devices can become part of MEC. They can be virtualized and contribute some computing/storage capacity to host application, platform, and infrastructure services. For this case, certain management and incentive mechanisms are needed to be implemented so users are willing to contribute part of their devices to host the services. Additional security and privacy schemes should also be deployed [33], [34].

IV. MEC Architectures

MEC provides a distributed computing environment to deploy applications and services in wireless access networks. The key element of MEC is MEC servers [18] which provide computing resources, storage, connectivity, and access to radio and network information. MEC servers can be embedded or colocated with the wireless networking elements such as APs, BSs, routers, switches, radio network controllers, and cell aggregation sites as illustrated in Fig. 1. In general, end user mobile devices/sensors may also become MEC servers and contribute some computing capacity for applications to run on them. Thus, MEC will enable applications and services to be hosted on top of these networks and end device elements, i.e., above the network layer, providing not only forwarding but in-network processing and caching of data.

It is desirable that MEC is generic and open to support various application/software services, as well as platform and infrastructure services. Not only can users request services and offload work, but the network can automatically and seamlessly perform services and process data according to the network conditions and contexts. An open management interface is important, through which third parties can lease resources and deploy new services. The architecture of such a smart computation-capable edge network is not the simple extension to traditional cloud architectures due to its distinctive characteristics and requirements such as heterogeneity, exposure to real-time radio and network information, and a variety of contextualized in-network services. A new MEC server and network architecture is critically needed.

A. MEC Server

Fig. 4 shows a general architecture and interfaces of MEC server platform as defined by ETSI MEC group [18]. The MEC server consists of a hosting infrastructure and a virtualized application platform. The hosting infrastructure contains hardware resources and a virtualization layer, whose details are abstracted from the applications. The MEC server has an infrastructure

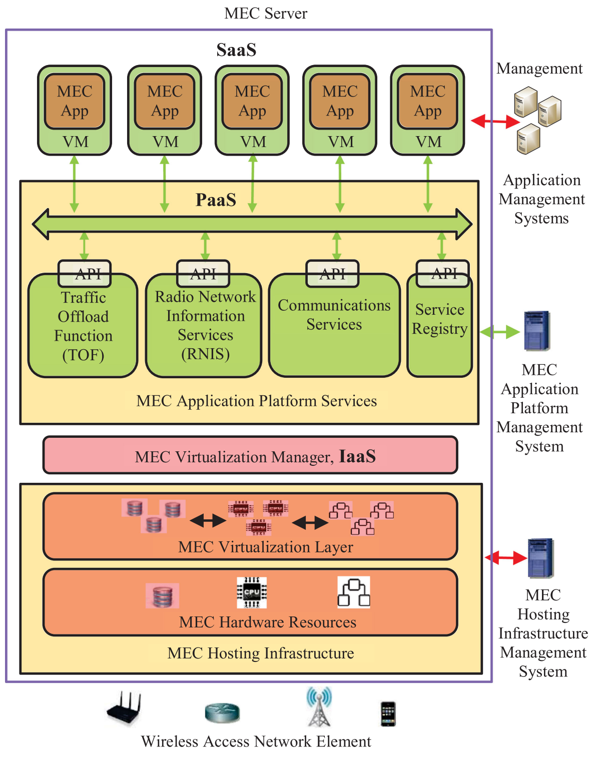
MEC Server
SaaS
MEC App
VM
PaaS
API
Traffic Offload Function (TOF)
Radio Network Information Services (RNIS)
Communications Services
Service Registry
MEC Application Platform Services
MEC Virtualization Manager, IaaS
MEC Virtualization Layer
MEC Hardware Resources
MEC Hosting Infrastructure
Management
Application Management Systems
MEC Application Platform Management System
MEC Hosting Infrastructure Management System
Wireless Access Network Element

Fig. 4. MEC server platform overview.

management interface through which an outside management system can create and manage on-demand computing instances or VMs on the MEC server. Each VM has certain system resource capacities such as CPU speed, number of cores, memory size, and storage capacity. Once VMs are created, they can be used to host applications and execute programs through the application platform. The MEC application platform offers the capabilities to host various applications, and consists of the application's virtualization manager and application platform services. The MEC virtualization manager provides a multi-tenant environment. MEC applications from network operators and third parties can be deployed and run within VMs. The MEC application-platform services provide a set of middleware services to the applications, including communications services, service registry, radio network information services (RNIS), and traffic offload function (TOF). The communication services enable the applications hosted on an MEC server to communicate with the application-platform services and with each other based on a publish-subscribe messaging protocol. The service registry contains the information of the services available on an MEC server, and allows applications to discover and locate the end-points for the services they require, and to publish their own service end-point for other applications to use. The MEC RNIS provide authorized applications with low-level radio and network information, such as cell ID, cell load, user location, network measurement and statistics information, etc. An application can select the RNIS information to subscribe and leverage the information for its own services and/or publish them for other applications. The TOF service prioritizes traffic and routes the selected user-data streams to and from authorized applications based on the policy. The traffic offloading policy sets the filters at the stream and packet levels. Applications may terminate the incoming data or process it and send back to the network. Security mechanisms will be required for authentication and authorization of service subscription, publishing, and access to the registry, as well as protection against malicious or misbehaving applications.

There is an application platform management interface through which the operator's management system can communicate with and manage the MEC application platform. The management interface is application agnostic and supports the functions such as configuration management of the application platform, life cycle of the applications, and VM operations and management. The configuration management of the application platform provides a standard way to package applications, including a descriptor of the application and system properties at the VM level relating to computing, storage and networking resource configuration, as well as access control mechanisms. It allows deployment of virtual appliances across multivendor MEC platforms, and ensures software integrity protection. Furthermore, the operators can use the application platform management interface to configure policy for the applications, and manage the life cycle of the applications such as deploy, start, stop, and undeploy. The MEC application platform management interface also supports fault management and performance measurements at the VM level. In addition to the MEC application platform management interface, there are application-specific management interfaces which provide means to manage the respective applications. The ETSI MEC group is working on its first release of the specification, mainly focusing on platform services and APIs.

There is a diverse set of technical challenges in design, deployment, and management of an MEC server. First, an MEC server is tightly integrated or collocated with a networking element. Its resources and capabilities may be limited. Efficient management framework and resource allocation mechanisms at the three layers, i.e., the hosting infrastructure, VM and application platform, and software applications and services, are important. Different schemes, such as a centralized controller or distributed controllers that communicate and cooperate to allocate and manage resources, need to be developed and evaluated. Second, a standard and flexible application platform management interface is needed for application portability so the applications can be seamlessly loaded and executed in the MEC platform that may be provided by different vendors. This will relieve the application developers from unnecessary burden in software development and integration, and allows for the rapid transfer of applications between MEC servers. Third, an MEC server should not only provide a secure sandbox for applications but also meet wireless network related security requirements. The platform should provide the mechanisms to ensure the isolation between the VMs, the security of platform software and hosted applications, as well as the security of communications. The applications can only access the infrastructure and platform resources, data, network information, and use traffic and context information for which they are authorized. The existing cloud computing virtualization technologies such as kernel-based or container-based can be applied to the MEC platform to solve certain security problems. However, there are some MEC-specific issues, for example, whether an application is allowed to access certain radio channel or mobile user information. Fourth, deployment of MEC servers should have a minimal impact to the performance, resiliency, and normal operation of wireless networks. The extra layers introduced by virtualization should not degrade performance, e.g., throughput, latency and packet loss, and incur minimal overhead for the hosted applications. Fault tolerance mechanisms may also be needed in the MEC platform to protect against any abnormality of the hosted software applications. The high-availability requirements of the network should be addressed, and in the event of a fault within the MEC platform or an MEC server failure, a failsafe mechanism should prevent it from adversely affecting the normal operation of the wireless network.

TABLE II
PROGRAMMING FRAMEWORKS FOR OFFLOADING

	Offloading Approach	Remote Execution Unit
Cloudlet	Dynamic VM Synthesis using the VM overlay from mobile device and the base VM in the cloudlet to create a customized environment for execution.	VM
MAUI	Partitioning of application program and offloading of certain methods to the cloud	Method
CloneCloud	Partitioning of application program and migration of certain threads to the cloud with application-level virtual machines	Thread

B. MEC Network Architecture

Multiple MEC servers can be deployed at different locations and hierarchy levels of a wireless access network, and each server may host a number of virtual application instances. MEC may be used to provide web services (WS) [35], [36] and other services with the traditional IP network architecture and TCP/IP protocol stack. To provide WSs, conventional URIs can be used to address web resources, and dynamic DNS allows to discover the WSs in the MEC network. Web proxies deployed in the MEC can be used to transparently intercept the user requests and direct the requests to the appropriate WSs. In addition, traditional service-oriented architecture (SOA) and WS interfaces (HTTP-based RESTful APIs or SOAP over HTTP) allow web applications to interoperate and perform mashups so that web applications can integrate their own data and functions with other WSs to create new services, and use content from multiple sources to produce enriched dynamic results through the published APIs. Clients can request WSs in the MEC and obtain mashup results from WSs through the standard web interface. Mashup presentation can be done with the client web browser that combines and reformats the data, or on a remote MEC server that integrates, analyzes, and reformats data and transmits the produced new data to the user's browser.

Computation offloading has recently attracted research attention, which improves performance and reduces the energy consumption of mobile devices. The challenges are to decide what to offload and when to offload with minimum programmer effort in a shared MEC environment. In [7], Satyanarayanan *et al*. proposed to use VM technology for transient customization of cloudlet infrastructure to support offloading. A mobile device transmits the customized VM overlay to the cloudlet, which applies it to the base VM to dynamically synthesize the launch VM. The mobile application is executed within a customized guest environment in the launch VM and brought to a state ready for user interaction. The device and the cloudlet each run a controller instance, which performs service discovery and network management. A secure TCP tunnel is established between the controller instances on the device and cloudlet using secure sockets layer for user authentication, overlay VM fetch, and application service launch and binding. After finishing the job, the post-use cleanup process ensures that cloudlet infrastructure is restored to its original software state without manual intervention. However, the VM synthesis time on the proof-of-concept prototype is in a range of 60–90 s. Two major components contributed to VM synthesis time are overlay transmission and decompressing and applying the overlay on the cloudlet, which need to be optimized such as using higher bandwidth for communications, caching VM images, and exploiting parallelism for decompression and overlay application in order to support many real-time applications.

MAUI [47] designs a platform that exploits .NET framework to dynamically partition mobile applications and offload the methods to the remote server for execution at runtime. It profiles the mobile device's energy consumption characteristics, the running time, and resource needs of individual methods, as well as the network bandwidth, latency, and packet loss, and determines which methods should be offloaded to the remote MAUI server and which should execute locally on the mobile device. In the implementation, the profiling and measurement collection of the application program's energy and data transfer requirements are done at the mobile device. The decision engine and optimization algorithms actually run on the remote server in order to save mobile's energy, which makes offloading decision using the collected information with an objective to minimize the mobile's energy consumption subject to latency constraints. The proxies and interfaces on the mobile device and MAUI server handles control message exchanges and data transfer for offloaded methods through remote procedure calls. The MAUI coordinator on the server handles authentication and resource allocation for incoming requests to instantiate execution of the methods offloaded from mobile devices.

CloneCloud [48] enables mobile applications running in an application-level VM to seamlessly offload part of their execution from mobile devices onto device clones operating in a computational cloud. The application is automatically partitioned. At runtime, a thread is migrated from the mobile device at a chosen point to the device clone in the cloud, executing in the cloud for the remainder of the partition until reaching a re-integration point. The thread then returns to the mobile device, along with remotely created state. It finally merges into the original process on the mobile device. Migration points are chosen to optimize the cost of the partitioned application such as execution time and energy use based on static analysis and dynamic profiling.

In comparison as shown in Table II, Cloudlets focus on dynamic synthesis and migration of VMs for remote execution of

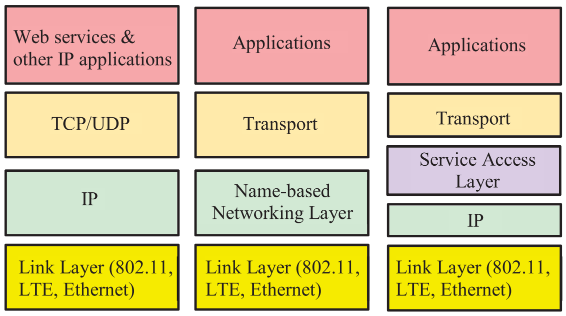
Web services & other IP applications
TCP/UDP
IP
Link Layer (802.11, LTE, Ethernet)
Applications
Transport
Name-based Networking Layer
Link Layer (802.11, LTE, Ethernet)
Applications
Transport
Service Access Layer
IP
Link Layer (802.11, LTE, Ethernet)

Fig. 5. Alternative network protocol stacks for MEC.

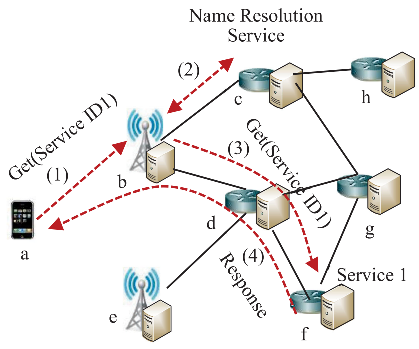

Fig. 6. Client *a* seeks to get service 1 hosted on MEC server *f*. *b* and *d* act as service routers to forward requests and responses. *b* uses name resolution service hosted on *c* to resolves the service location based on the requested service ID and then forwards Get(Service ID1) request to server *f* via router *d*.

offloaded applications in a guest environment. MAUI focuses on partitioning and profiling application programs and offloading some methods to optimize mobile energy consumption and performance. The remote execution unit and the granularity of state transfer in MAUI are at the method level. CloneCloud uses application-layer VMs and its migration operates at the granularity of a thread.

Although the TCP/IP protocol suite has been remarkably successful for supporting various applications, it was designed for services and technology assumptions that may not be well-matched to mobile devices and information-oriented services that the MEC targets at. For example, IP address assignments are designed to be relatively static and service request requires to determine the host's IP address, but a mobile user cares about the service itself, regardless of the specific physical location where the service resides. TCP assumes a continuous end-to-end path, which is often violated in mobile scenarios. Traditional mobile IP techniques introduce extra latency and overhead. Network address (NA) translation fragments the network and needs gateways and servers, which causes extra burden for both developers and end-users [37], [40]. The challenges include system and protocol optimization by taking into consideration the underlying wireless communications architecture, developing better service and data placement, resource allocation, and computation orchestration schemes to improve efficiency and reduce latency and overhead, as well as simplify management.

Several network layer solutions for service discovery and interoperation have been proposed to improve system performance [6], [9], [10], [38], [39]. These MEC architecture designs are based on name-oriented future Internet architectures [37], [40] that enables applications to communicate directly on service names or IDs (SIDs) and promise seamless mobility handling as well as more efficient data and service access. However, as shown in Fig. 5, they require new network protocol stacks running on MEC servers and user devices. These future Internet stacks are still in the lab research phase, and their performance compared to the traditional TCP/IP stack is still unknown in realistic large deployments.

Based on a name-oriented future Internet architecture MobilityFirst [37], an MEC platform is designed [9], [10], in which routers may have an optional cloudlet to host services, and any network-connected object including mobile device, router, data chunk, multicast group, context, or service is given a globally unique identifier (GUID). A GUID is a flat name, e.g., a public key of an object for authentication purpose or simply the hash digest of the human readable name of an object, which is assigned by a distributed name certification service. A network service may run on multiple nodes at different locations, serving the users that are often mobile and multihomed. GUID is independent of the server and user address. By decoupling identity from location at the network layer, the name-oriented networking paradigm has potential to address the inefficiency of IP networks in supporting dynamic service access by mobile users. A global name resolution service (GNRS) provides the dynamic binding between GUIDs and their current NAs or network attachment points, and an object dynamically registers its location(s), i.e., NAs with GNRS. GNRS employs a logically centralized, but physically distributed directory service to track each object's current NA. Each service deployed in the network is thus identified by its GUID, called SID, and its location(s) are registered with GNRS. As shown in Fig. 6, a user can get one or more services with *get(SID1, SID2, . . .)*, or request one or more service to be performed on a data chunk it sends with *send(srcGUID, dstGUID, SID1, SID2 . . . , payload)*. The routers will query GNRS to obtain the NAs corresponding to the current points of attachment of the requested services and the destination GUID. The routers will forward the data chunk to the services, and then deliver the processed chunk to the destination using NAs rather than GUIDs along the path with a routing protocol. The forwarding process is based on hop-by-hop block transfer of data chunks between routers, and supports the concept of late binding by which routers optionally rebind the GUID to a set of NAs to deal with disconnection, mobility, server failure, virtual service migration, and so on. Transparent services are activated by the network without being known by the sender and the receiver. A router can apply some preconfigured rules, e.g., a specified source/destination GUID or a specific pattern in the chunk header or payload, to intercept selected chunks and forward them to a transparent service. This approach enables several flexible communications and service access mechanisms, for example, multiple service replicas for load balancing, automatic server failover, virtual service migration, and user mobility handling.

In [6], an MEC architecture, called CloudEdge, is proposed, which integrates name-oriented networking and software-defined networks (SDN) with decoupled data and control planes. In contrast to the above GNRS that only provides the mapping between GUIDs and NAs, a logically centralized and physically distributed controller is used to program the data plane for not only data forwarding but also content processing and caching. An open interface allows network administrators to abstract and manage the underlying resources and platform according to dynamic user demands and wireless network conditions. In-network processing, storage, and communications are integrated under a unified software-defined platform to fully exploit the potential of virtualization to enhance user experience and optimize service orchestration in the network. A service may expose a set of operations by publishing its interface with the controller. Data forwarding is treated as one of the services. User requests are routed to the appropriate virtual service instances with a service-aware routing algorithm, and resources are dynamically allocated based on the policy. The controller interacts with the data plane to set the processing rules on demand at the responsible servers for data computation and caching, configure the filtering and forwarding rules for data forwarding at each intermediate router along the path, and monitor and manage the resources and platform. The network data plane then executes the data forwarding and computation services, and returns the results to the requester(s) or destination(s). Transparent services are supported by allowing the network to invoke the services based on the preset rules by the controller. A separate management plane is used for service deployment, monitoring, migration, scaling, and other management functions.

A named function networking (NFN) architecture has been introduced in [38] and [39], which extends content centric networks (CCNs) [40] to resolve names to computation functions. Borrowing some concepts from active networking [41]–[43], a user can send an interest to the network with a function expression or a program that includes code and data. The network returns the user the computed results. The functional programs are expressed in λ-calculus [44], [45], encoded in the CCN names that are hierarchical and aggregable, and carried in the Interest packet. The program is executed in two levels: the name-manipulation and the orchestration of computation distribution with λ-calculus, and the native code function execution for actual computation and data processing tasks at the identified execution nodes.

An NFN node integrates a λ-calculus expression resolution engine, and optionally hosts an application processing /execution environment or compute server. CCN employs name-based flooding model to enable a source to publish its reachability to a named data object, and establish an entry in the forward information base of CCN routers. The router uses the longest-match name lookup to forward Interests. Similarly, if an NFN node hosts a compute service, its named function is published as a λ-calculus expression in the network, and interests can be forwarded to the function host. For example, a user sends an interest to request a transcoded version of a media content object [37], $i[\lambda y.(/name/of/transcoder\ \ y)\ \ /name/of/media\ \}$. The interest is forwarded toward the data source of */name/of/media* using the longest prefix matching rule. If a copy of the transcoded version of requested media is cached at a node along the path to the data source of */name/of/media*, the result will be returned the requesting user along the reverse path. Otherwise, one of the NFN nodes has received the interest on the path to the data source and may attempt to compute the result based on the policies and processing resource availability. The node can extract the names from the λ-expression in the interest, and fork separate individual Interests, e.g., for */name/of/media* and */name/of/transcoder*. If the video data and the transcoder code are retrieved, the node will then compute and return the result. The result may be cached for serving other requests. If any of the interests does not yield the content back, e.g., because the code cannot be distributed to the other node, the NFN node may become a computation-proxy that forwards the interest toward the code source, i.e., pushing computation toward the code source. If the computation is served along the way to the code source, the result will travel the reverse route back to the proxy point and then back to the original requester. In general, an NFN node can evaluate and split a named expression, and retrieve code, data, intermediate results, and thunks, or distribute the tasks by sending the interests with subexpressions to different nodes. It thus finishes the computation and sends the response to the original requester. The thunks enable the evaluation of a named expression with asynchronous and parallel computations.

In addition, an architecture, called Serval, is presented in [46], which designs a service access layer (SAL) sitting between the transport and network layers that resolves the service ID and demultiplex flows as shown in Fig. 5(c). It enables applications to communicate directly on service names through active socket binding with SAL. A service name or serviceID corresponds to a group of instances offering the same service. The serviceID uses a 256-bit hierarchical namespace, such as the IPv6 format. Additionally, the host-local flow IDs are used at the end hosts to indicate the flows associated with a socket, and NAs are used to identify the host interfaces. The SAL maps serviceID in packets to NAs based on rules in its service table. A packet may be forwarded to next hop by a service router, demultiplexed and delivered to a local socket for service at a server, delayed, or dropped. The longest prefix matching is used for service resolution and forwarding. The service table can be programmed through the local service controller, and different service discovery algorithms, e.g., serviceID prefix dissemination, distributed hash table, service request flooding, lookup with name resolution servers, can be used to populate the service forwarding table. An active socket allows an application to bind to a socket and acts on the events at the hosts. The SAL can establish multiple flows over different interfaces or paths to a remote end-point. The transport layer is unaware of flow identifiers and interface addresses. The SAL can migrate a flow from one address, interface, or path to another without breaking up service access, which allows to support client mobility, interface failover, and VM migration.

An MEC PaaS programming model, termed Mobile Fog, is presented in [31] to support geospatially distributed IoT applications such as smart traffic control, video surveillance, and vehicle identification and tracking. Mobile Fog provides a programming abstraction and supports applications dynamically scaling at runtime. The tasks or processes of an application are mapped onto distributed computing instances in the network as well as various edge devices. Mobile Fog runtime platform provides APIs through which applications can call a set of functions. The applications also implement a set of event handlers

TABLE III
OVERVIEW OF MEC SYSTEM ARCHITECTURES

MEC architecture	Service model	Targeted layer	Service naming	Service discovery, resolution and routing	Dynamic service programming and user program loading
Applications with Traditional TCP/IP	SaaS	Application	URI	Dynamic DNS and IP routing	Clients can send the program to web services. Web services execute the program and perform mashups.
MobilityFirst	SaaS	Network and transport	Flat service ID	Resolve serviceID to network address with distributed name resolution servers	Assume that the service instances have been deployed in the network. Support late binding of serviceID to addresses, and service location migration.
CloudEdge	SaaS	Network, transport, and application	Flat service ID	Program the data plane with the central controller	A unified controller can automatically program packet forwarding rules, content caching, and in-network service policies and parameters based on real-time network information. Support dynamic service scaling and service instance migration.
NFN	SaaS	Network	λ-calculus expression	Service name prefix dissemination	Users load the program (code + data). Network layer distributes tasks based on function names, and computing nodes execute the code.
Serval	SaaS	Service sublayer between network and transport	Hierarchical service ID	Resolve serviceID to network address using local service table. Support different algorithms to populate the service table.	Assume that the service instances have been deployed in the network. Support late binding of serviceID to address, and service location migration.
Mobile Fog	PaaS	Application	Unique ID (format not specified)	Tree based and point-to-point task communications depending on the lower layer	Support dynamic scaling and on-demand creation of service instances. Provide a management interface for service deployment.

that are invoked by the Mobile Fog runtime system upon certain events. Mobile Fog provides the information about the capabilities, resources, geophysical location, and network hierarchy level of underlying physical devices in order to allow application code to perform location and context-aware processes. Running processes on different devices can communicate with each other using communication APIs. Each process image can be deployed with an associated unique identifier, called appkey, with which the application can be managed using the management interfaces provided by Mobile Fog. When a computing instance becomes overloaded, Mobile Fog transparently creates on-demand instances at the same network hierarchy level as the overloaded instance, and distributes the workload over multiple computing instances.

To get a deeper insight into the salient features of the MEC architectures, we present an overview in Table III. The service model in the table indicates what kind of services the architecture is for. The network stack layer(s) that each solution mainly targets at is described in the network layer column of the table. The service naming indicates how the service is named in the architecture, a flat name, e.g., a public key for self-certifying purpose, or a hierarchical name for aggregation purpose. The service discovery, resolution, and routing mechanisms in each architecture are summarized in the table. Some proposed architectures, e.g., NFN, allow users to dynamically load the program (code along with data) through data plane and execute the code in the network. The routers in NFN have richer programming primitive to distribute the tasks along the path based on λ-calculus function name expression, and place the functions to be executed at a site. Mobile Fog is a PaaS platform with dynamic scaling support, and creates on-demand service instances at the same network hierarchy level for load balancing as a computing instance becomes overloaded. The service or application tasks can be deployed by the users using a management plane and APIs. CloudEdge uses a unified controller to automatically program packet forwarding rules, content caching, and in-network service policies and parameters based on real-time network information, and support dynamic service scaling and service instance migration. Other architectures, such as MobilityFirst and Serval, focus on service discovery and service request routing and forwarding, and support late binding and service location migration under the assumption that the service instances have been deployed in the network. Traditional WSs and SOA with TCP/IP support mashup and dynamic content creation at the application layer. Clients may send the programs to one or more WSs that execute the programs and integrate their own resources, application, and data with external data and functions from other WSs to produce rich results and deliver to the clients.

V. OPEN RESEARCH CHALLENGES

Despite its promises and advantages as discussed above, the realization of MEC systems faces a new set of fundamental challenges that need to be addressed.

A. MEC Architecture to Support Diverse Services

The architectures surveyed in this paper represent the initial attempts to design the MEC systems that integrate computing and storage capabilities in wireless edge networks. However, many technical issues in architecture design, some of which are very fundamental, have still not been well addressed. A wireless access network infrastructure of compute, storage, and networking devices itself breaks the traditional Internet layers principle with which routing is a layer 3 function and routers should not handle the functions above layer 3. The design issues such as how to partition the functions between the network layer and application layer and how to partition computing and communication need to be carefully analyzed and understood, as well as their impacts to the system performance need to be thoroughly evaluated. For example, should a network provide dynamic programmability directly to users and allow users sending "code" in the request or data messages to manipulate the forwarding process of the messages in the network? If so, where should this functionality be placed? Should it be part of the data forwarding plane at the network layer or engineered on top of it?

Active networking research in the past [36], [39], [40] envisioned that users could load programs into the network, and the network would provide richer programming primitives. Functions, input parameters, and output results can be carried in the packet header, the packet body, or both. NFN employs λ-calculus expression to inject more complex logic into network, and function names are carried in the data packet headers as part of network layer processing. Furthermore, it is also possible to carry the routing logics associated with real-time network information, e.g., Request (Service_2 if Service_1 is busy), in the packet header. This approach will let the network make forwarding decisions for en-route distribution of computation tasks. Actually, the naming scheme as well as routing protocol determines how a computation job is executed in the network, i.e., how a name is decomposed into individual computation tasks that will be routed to the servers in the network, and which servers are involved in the execution of computation tasks.

Alternatively, the logic associated with functions and input parameters can be implemented at the application layer, similar to WSs. For example, the user program can be forwarded to a platform service and get executed, where the platform service may distribute computation tasks to and coordinate with other services for task execution, and retrieve data, code, and intermediate results to compute the final result. Clearly, each of these approaches has its pros and cons. Network layer realization of more complex logics for orchestrating and distributing data computations certainly increase routers' complexity and introduce execution performance overhead. It also causes security concerns (the advanced technical survey and exploration of the MEC security is part of our future work). On the other hand, the network-layer approach may yield better service routing and task distribution decisions. Note that unlike active networks, the MEC users intend to obtain services, instead of explicitly modifying the operation of the network and programing a data path. Given the recent advances in virtualization, computing, and SDN networking technologies, network programing models may need to be revisited to gain new understanding of the current technological limits and the needs of emerging application scenarios that push the limits and demand transformation of today's wireless infrastructures. The quantitative performance comparison is needed to determine the design choice and the tradeoffs to partition the functions between the network layer and application layer.

In addition, the pros and cons of running MEC on traditional TCP/IP protocol stack vs. modifying the TCP/IP protocol stack should be analyzed and evaluated further. Certainly, MEC running traditional TCP/IP protocol stack will simplify deployment and allow backward compatibility with existing mobile devices and networking devices such as routers and switches. Research is needed to optimize systems and protocols such as in-network service deployment, service discovery, communications, orchestration, and resource management by considering the underlying wireless communications architecture and application requirements. The modifications of TCP/IP protocols, adding a new layer, or even clean-slate designs in the new MEC architectures need to be justified, and their benefits and disadvantages should be evaluated and quantified through actual experimental testing. Research is needed to develop and examine the mechanisms for using clean-slate architectures to address scalability, reduce latency, and enhance performance.

Furthermore, the MEC architectures are expected to support diverse application and service models, including user requested services, transparent services, third-party services, IaaS, PaaS, and SaaS, as well as transient traffic (the data that do not need in-network services), in an efficient, robust, and scalable way, according to the technical and business needs of the operators. Most of the existing designs focus on supporting one or a few of the service models. They need to be enhanced to provide a holistic solution. Alternative architectures may also be designed to support more features and achieve better performance.

The performances of all architecture designs, including the existing ones and new designs, should be compared under various scenarios through realistic prototyping and experiments to gain deep insights into the tradeoffs of alternative approaches and techniques, as well as their impacts to the efficiency, overhead, complexity, latency, and throughput performance. The promise of MEC architectures needs to be verified through experimental research to prove that they can support the challenging application requirements, e.g., the low latency in millisecond order with a very high density of devices (millions of smart devices), a large volume of aggregate traffic (gigabytes), and a wide range of service characteristics (traffic control, video streaming, security monitoring, etc.), in massive IoT deployment, all sharing a common MEC infrastructure and networking/computation framework.

B. Efficient Platform With Open APIs

The platform and the application-specific functions should be decoupled. MEC devices are heterogeneous, but the MEC platform is desired to be generic and open, and provides a consistent execution environment so that various distributed applications can be easily developed and run on the top of it through a standard API, independent of the underlying infrastructure. The API will facilitate the interactions between the applications and the underlying infrastructure. The platform should provide not only conventional services such as task communications, event handling, service registry, data forwarding, but also wireless access network information such as radio channel state, traffic load, user location, network measurement and statistics information, etc. In addition, the abstraction is important and challenging due to heterogeneity. The platform may need to provide information about the capabilities, resources, geophysical location, and network hierarchy level of underlying physical devices in order to allow application programs to perform location and context-aware processes, but certain abstraction is required for application development, manageability, and scalability. Efficient and flexible network virtualization and dynamic scaling at runtime should be supported by the platform for deploying different applications. Furthermore, security mechanisms are required for access to the services and protection against malicious or misbehaving applications.

Platform-independent programming languages are preferable to implement the application functions. However, different MEC servers or VMs may offer different execution environments, e.g., some support Python whereas others support Java VMs with droplets. The programming model and service routing algorithms require taking into consideration these potential heterogeneous execution environments.

C. Network Control Protocols and Algorithms

To build such a smart MEC system, new network control protocols and algorithms are critically needed, which in turn poses severe challenges to the design. The key challenges include: how to quickly and efficiently route the requests and steer the traffic to the best service instances in the network, and provision the services, how to seamlessly handle user mobility and corresponding services, as well as dynamically scale a slice to enable adaptive allocation of the computation functions and associated data close to the moving users, how to seamlessly migrate a service from one virtual network to another and keep the session states during migration to avoid service interruption, and how to realize efficient distributed service control?

Conventional routing is aimed to forward data to the destination, or forward a request to the nearest content source and return the data to the requester. However, in MEC networks, the routing is to achieve in-network computation and obtain the result. For example, if the computation task is to aggregate IoT data for analysis, the routing strategy should then route the data to achieve a better aggregation performance.

D. Resource Management and Computation Orchestration

In contrast to dedicated data centers with well-planned devices and networks, MEC networks are expected to consist of highly dynamic and heterogeneous multidimensional resources (CPU, storage, and bandwidth) at geospatially distributed points and different levels of network hierarchy. In addition, the fact that mobile devices often change their points of attachment to the network with unreliable wireless links greatly complicates the underlying resource allocation and scheduling algorithms. The strategies and algorithms to deploy the services at appropriate locations in the network need to be developed. The schemes that assign computation tasks according to dynamically available computation resources as well as efficient resource management and task scheduling algorithms need to be investigated to optimize system performance by taking into consideration application requirements, subtask synchronization requirements, resource constraints, and tradeoffs between data shipping and processing. New mechanisms to virtualize the heterogeneous network and to rapidly provision and release resources as needed in the distributed heterogeneous devices are required.

E. Interplay With Traditional Clouds and Mobile Devices

The execution of many applications may require the tasks to span across mobile end devices, MEC network nodes, and distant central clouds. Furthermore, these applications can involve several stages of computation, with data flowing from one stage to another, in a pipelined fashion. As a result, it is important to develop a set of carefully designed runtime management strategies to ensure efficient execution of these parallel tasks, otherwise, the application's low-latency requirements may not be able to be satisfied. Innovative runtime management schemes, which efficiently distribute jobs and orchestrate the parallel distributed execution of tasks in such a heterogeneous environment with real-time constraints, need to be developed.

VI. Conclusion

This paper presents a survey on the architectures and designs of MEC systems from a technical point of view, and outlines the existing solutions as well as open challenges and future research directions. The characteristics of MEC are first explored and the MEC applications are classified based on different criteria. The service models and deployment scenarios are categorized, and the factors influencing the design of MEC platform are discussed. The architectures and designs of MEC systems are then surveyed, and the technical issues, existing solutions, and approaches are presented. The challenges and future research directions of MEC are discussed. As part of our future work, the issues and techniques regarding the MEC security will be surveyed and explored.

References

[1] Z. Sheng, C. Mahapatra, C. Zhu, and V. C. M. Leung, "Recent advances in industrial wireless sensor networks toward efficient management in IoT," *IEEE Access*, vol. 3, pp. 622–637, Jun. 2015.

[2] C. Zhu, V. C. M. Leung, L. Shu, and E. C.-H. Ngai, "Green Internet of Things for smart world," *IEEE Access*, vol. 3, pp. 2151–2162, Nov. 2015.

[3] B. Chandramouli, J. Claessens, S. Nath, I. Santos, and W. Zhou, "Race: Real-time applications over cloud-edge," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2012, pp. 625–628

[4] A. Chandra, J. Weissman, and B. Heintz, "Decentralized edge clouds," *IEEE Internet Comput.*, vol. 17, no. 5, pp. 70–73, Sep./Oct. 2013.

[5] U. Drolia *et al.*, "The case for mobile edge-clouds," in *Proc. 2013 IEEE 10th Int. Conf. Ubiquitous Intell. Comput. 2013 IEEE 10th Int. Conf. Auton. Trusted Comput.*, 2013, pp. 209–215.

[6] H. Liu and K. Smith, "Improving the expected quality of experience in cloud-enabled wireless access networks," in *Proc. 2015 IEEE 12th Int. Conf. Mobile Ad Hoc Sens. Syst.*, Oct. 2015, pp. 519–524.

[7] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct./Dec. 2009.

[8] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *Proc. IEEE Symp. Comput. Commun.*, 2012, pp. 000059–000066.

[9] Y. Chen, B. Liu, Y. Chen, A. Li, X. Yang, and J. Bi, "Packetcloud: An open platform for elastic in-network services," in *Proc. 8th ACM Int. Workshop Mobility Evolving Internet Archit.*, 2013, pp. 17–22.

[10] F. Bronzino *et al.*, "In-network compute extensions for rate-adaptive content delivery in mobile networks," in *Proc. 2014 IEEE 22nd Int. Conf. Netw. Protocols*, 2014, pp. 511–517.

[11] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the Internet of Things,” in *Proc. Workshop Mobile Cloud Comput.*, 2012, pp. 13–16.
[12] “Fog computing aims to reduce processing burden of cloud systems.” [Online]. Available: http://www.eweek.com/
[13] J. Zhu, D. S. Chan, M. S. Prabhu, P. Natarajan, H. Hu, and F. Bonomi, “Improving web sites performance using edge servers in fog computing architecture,” in *Proc. 2013 IEEE 7th Int. Symp. Serv.-Oriented Syst. Eng.*, 2013, pp. 320–323.
[14] L. M. Vaquero and L. Rodero-Merino, “Finding your way in the fog: Towards a comprehensive definition of fog computing,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, pp. 27–32, 2014.
[15] S. Yi, C. Li, and Q. Li, “A survey of fog computing: Concepts,” in *Proc. 2015 Workshop Mobile Big Data*, Jun. 2015, pp. 37–42.
[16] I. Stojmenovic and S. Wen, “The fog computing paradigm: Scenarios and security issues,” in *Proc. 2014 Federated Conf. Comput. Sci. Inf. Syst.*, 2014, pp. 1–8.
[17] H. Madsen, G. Albeanu, B. Burtschy, and F. Popentiu-Vladicescu, “Reliability in the utility computing era: Towards reliable fog computing,” in *Proc. 2013 20th Int. Conf. Syst., Signals Image Process.*, 2013, pp. 43–46.
[18] “Mobile edge computing,” ETSI White Paper. [Online]. Available: https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing_-_introductory_technical_white_paper_v1%2018-09-14.pdf
[19] S. Nunna *et al.*, “Enabling real-time context-aware collaboration through 5G and mobile edge computing,” in *Proc. 12th Int. Conf. Inf. Technol.—New Gener.*, Apr. 2015, pp. 601–605.
[20] G. Orsini, D. Bade, and W. Lamersdorf, “Computing at the mobile edge: Designing elastic android applications for computation offloading,” in *Proc. 8th IFIP Wireless Mobile Netw. Conf.*, 2015, pp. 112–119.
[21] N. Takahashi, H. Tanaka, and R. Kawamura, “Analysis of process assignment in multi-tier mobile cloud computing and application to edge accelerated web browsing,” in *Proc. 3rd IEEE Int. Conf. Mobile Cloud Comput., Serv., Eng.*, Mar. 2015, pp. 233–234.
[22] ETSI, Industry specification group (ISG) for mobile-edge computing (MEC). [Online]. Available: http://www.etsi.org/technologies-clusters/technologies/mobile-edge-computing
[23] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, “A survey of mobile cloud computing: Architecture, applications, and approaches,” *Wireless Commun. Mobile Comput.*, vol. 13, pp. 1587–1611, 2013.
[24] Y. Wang, I. Chen, and D. Wang, “A survey of mobile cloud computing applications: Perspectives and challenges,” *Wireless Personal Commun.*, vol. 80, no. 4, pp. 1607–1623, Feb. 2015.
[25] L. Yang, J. Cao, S. Tang, T. Li, and A. Chan, “A framework for partitioning and execution of data stream applications in mobile cloud computing,” in *Proc. IEEE 5th Int. Conf. Cloud Comput.*, Jun. 2012, pp. 794–802.
[26] S. Sardellitti, G. Scutari, and S. Barbarossa, “Joint optimization of radio and computational resources for multicell mobile-edge computing,” *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.
[27] L. Zuo, L. Shu, S. Dong, C. Zhu, and Z. Zhou, “Dynamically weighted load evaluation method based on self-adaptive threshold in cloud computing,” *ACM Mobile Netw. Appl.*, vol. 21, no. 1, pp. 1–15, Jan. 2016.
[28] L. Zuo, S. Dong, L. Shu, C. Zhu, and G. Han, “A multiqueue interlacing peak scheduling method based on tasks’ classification in cloud computing,” *IEEE Systems J.*, vol. 12, no. 2, pp. 1518–1530, Jun. 2018.
[29] M. Beck, M. Werner, S. Feld, and T. Schimper, “Mobile edge computing: A taxonomy,” in *Proc. 6th Int. Conf. Adv. in Future Internet*, Lisbon, Portugal, Nov. 2014, pp. 48–54.
[30] A. Ahmed and E. Ahmed, “A survey on mobile edge computing,” in *Proc. 10th IEEE Int. Conf. Intell. Syst. Control*, 2016, pp. 1–8.
[31] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwalder, and B. Koldehofe, “Mobile fog: A programming model for large-scale applications on the internet of things,” in *Proc. 2nd ACM SIGCOMM Workshop Mobile Cloud Comput.*, 2013, pp. 15–20.
[32] I. Stojmenovic, “Fog computing: A cloud to the ground support for smart things and machine-to-machine networks,” in *Proc. 2014 Austral. Telecommun. Netw. Appl. Conf.*, 2014, pp. 117–122.
[33] C. Dsouza, G.-J. Ahn, and M. Taguinod, “Policy-driven security management for fog computing: Preliminary framework and a case study,” in *Proc. 2014 IEEE 15th Int. Conf. Inf. Reuse Integr.*, 2014, pp. 16–23.
[34] S. Yi, Z. Qin, and Q. Li, “Security and privacy issues of fog computing: A survey,” in *Proc. Workshop Autom. Software Archit.*, Aug. 2015, pp. 685–695.
[35] W3C, Web Services Architecture. [Online]. Available: http://www.w3.org/TR/ws-arch/
[36] A. Klein, C. Mannweiler, J. Schneider, and H. D. Schotten, “Access schemes for mobile cloud computing,” in *Proc. 11th Int. Conf. Mobile Data Manage.*, May 2010, pp. 387–392.
[37] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani, “Mobilityfirst: A robust and trustworthy mobility-centric architecture for the future internet,” *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 16, no. 3, pp. 2–13, 2012.
[38] M. Sifalakis, B. Kohler, C. Scherb, and C. Tschudin, “An information centric network for computing the distribution of computations,” in *Proc. 1st Int. ACM Conf. Inf. Centric Netw.*, Sep. 2014, pp. 137–146.
[39] C. Tschudin and M. Sifalakis, “Named functions and cached computations,” in *Proc. Annu. IEEE Conf. Consum. Commun. Netw.*, Jan. 2014, pp. 851–857.
[40] V. Jacobson *et al.*, “Networking named content,” in *Proc. 5th Int. Conf. Emerging Netw. Exp. Technol.*, 2009, pp. 1–12.
[41] S. Merugu *et al.*, “Bowman and CANEs: Implementation of an active network,” in *Proc. 37th Allerton Conf. Commun., Control Comput.*, Sep. 1999, pp. 147–156.
[42] M. Hicks *et al.*, “PLAN: A packet language for active networks,” in *Proc. 3rd ACM SIGPLAN Int. Conf. Funct. Program.*, 1998, pp. 86–93.
[43] A. Campbell *et al.*, “A survey of programmable networks,” *SIGCOMM Comput. Commun. Rev.*, vol. 29, no. 2, pp. 7–23, Apr. 1999.
[44] J.-L. Krivine, “A call-by-name lambda-calculus machine,” *Higher Order Symbol. Comput.*, vol. 20, no. 2, pp. 199–207, Sep. 2007.
[45] B. Pierce, *Types and Programming Languages*. Cambridge, MA, USA: MIT Press, 2002.
[46] E. Nordstrom *et al.*, “Serval: An end-host stack for service-centric networking,” in *Proc. 9th USENIX Symp. Netw. Syst. Design Implementation*, 2012, pp. 7–7.
[47] E. Cuervo *et al.*, “MAUI: Making smartphones last longer with code offload,” in *Proc. Proc. 8th Int. Conf. Mobile Syst., Appl., Serv.*, Jun. 2010, pp. 49–62.
[48] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, “CloneCloud: Elastic execution between mobile device and cloud,” in *Proc. 6th Eur. Conf. Comput. Syst.*, Apr. 2011, pp. 301–314.

Hang Liu (M’98–SM’05) received the Ph.D. degree in electrical engineering from the University of Pennsylvania, Philadelphia, PA, USA.

He joined the Catholic University of America (CUA), Washington, DC, USA, as an Associate Professor with the Department of Electrical Engineering and Computer Science, in 2013. Prior to joining CUA, he had more than ten years of research experience in networking industry, and worked in senior research and management positions at several companies. He also led several industry–university collaborative research projects. He was an Adjunct Professor of WINLAB, Department of Electrical and Computer Engineering, Rutgers University, from 2004 to 2012. He was an active participant in the IEEE 802 wireless standards and 3GPP standards. He was also the Rapporteur of a 3GPP work item. His research interests include wireless communications and networking, Internet of Things, mobile computing, future Internet architecture and protocols, content distribution, video streaming, and network security.

Dr. Liu was the Editor of the IEEE 802.11aa Standard and was the recipient of an IEEE Recognition Award.

Fahima Eldarrat is working toward the Ph.D. degree in computer science at the Catholic University of America, Washington, DC, USA.

Hanen Alqahtani is currently working toward the Ph.D. degree in computer science at the Catholic University of America, Washington, DC, USA.

She is also a Lecturer with Al-Majmaah University, Al Majmaah, Saudi Arabia.

Alex Reznik (M'94–SM'10) received the B.S.E.E. (*summa cum laude*) from Cooper Union, New York, NY, USA, the S.M. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA, and the Ph.D. degree in electrical engineering from Princeton University, Princeton, NJ, USA.

He is currently a Solution Architect with Hewlett Packard Enterprise's Telco Strategic Account Team, New York, NY, USA. In this role, he is involved in various aspects of helping a tier 1 telco evolve towards a flexible infrastructure capable of delivering on the full promises of 5G. Prior to May 2016, he was a Senior Principal Engineer with InterDigital, leading the company's research and development activities in the area wireless internet evolution. Since joining InterDigital in 1999, he has been involved in a wide range of projects, including leadership of 3G modem ASIC architecture, design of advanced wireless security systems, coordination of standards strategy in the cognitive networks space, development of advanced IP mobility and heterogeneous access technologies, and development of new content management techniques for the mobile edge. He held a visiting faculty appointment with WINLAB, Rutgers University, where he collaborated on research in cognitive radio, wireless security, and future mobile Internet. He is an inventor of more 130 granted U.S. patents.

Dr. Reznik was the recipient of numerous awards for Innovation at InterDigital. He served as the Vice-Chair of the Services Working Group at the Small Cells Forum.

Xavier de Foy received the Master of Engineering degree from the École Supérieure d'Électricité (Supélec), Gif-sur-Yvette, France, in 1994.

He is an Engineer with more 20 years of experience in networking and software systems in defense and telecommunication industries. Since 2007, he has been with InterDigital, Inc., Montréal, QC, Canada. His research interests include edge computing, distributed systems, and network architecture.

Yanyong Zhang (M'02–SM'15) received the Ph.D. degree from Pennsylvania State University, State College, PA, USA.

She is currently a Professor with the Electrical and Computer Engineering Department, Rutgers University, New Brunswick, NJ, USA. She is also a member of the Wireless Information Networks Laboratory (Winlab). She has authored or co-authored more than 100 technical papers in various international conferences and journals. Her current research interests include future Internet and pervasive computing.

Dr. Zhang is a member of the ACM and an Associate Editor for the IEEE TRANSACTIONS ON MOBILE COMPUTING AND SERVICE COMPUTING.