Journal of Network and Computer Applications 62 (2016) 18–40

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Review

Cloudlet deployment in local wireless networks: Motivation, architectures, applications, and open challenges

Usman Shaukat [a], Ejaz Ahmed [b], Zahid Anwar [a], Feng Xia [c,*]

[a] System Research Group, School of Electrical Engineering and Computer Science, National University of Science & Technology, Islamabad, Pakistan
[b] Department of Computer System & Technology, University of Malaya, 50603 Kuala Lumpur, Malaysia
[c] School of Software, Dalian University of Technology, Dalian 116620, China

ARTICLE INFO

Article history:
Received 22 February 2015
Received in revised form
24 September 2015
Accepted 6 November 2015
Available online 23 December 2015

Keywords:
Mobile Cloud Computing
Cloudlets
Mobile Device Augmentation
Local Wireless Networks

ABSTRACT

In past few years, advancement in mobile applications and their integration with Cloud computing services has introduced a new computing paradigm known as Mobile Cloud Computing. Although Cloud services support a wide range of mobile applications, access to these services suffers from several performance issues such as WAN latency, jitter, and packet losses. Cloudlet frameworks are proposed to overcome these performance issues. More specifically, Cloudlets aim to bring the Cloud or a specific part of the Cloud closer to the mobile device by utilizing proximate computing resources to perform compute-intensive tasks. This paper presents a comprehensive survey on the state-of-the-art mobile Cloudlet architectures. We also classify the state-of-the-art Cloudlet solutions by presenting a hierarchical taxonomy. Moreover, the areas of Cloudlet applications are also identified and presented. Cloudlet discovery, resource management, data security, mobility, application offloading, and most importantly incentives to deploy a Cloudlet are areas that still needs to be investigated by the research community. The critical aspects of the current Cloudlet frameworks in Mobile Cloud Computing are analyzed to determine the strengths and weaknesses of the frameworks. The similarities and differences of the frameworks based on the important parameters, such as scalability, mobility support, Internet dependency, dynamic configuration, energy savings, and execution cost, are also investigated. The requirements for deploying the Cloudlet in a Local Wireless Network are also highlighted and presented. We also discuss open research challenges that Cloudlet deployments face in Local Wireless Networks.
© 2015 Elsevier Ltd. All rights reserved.

Contents

* Corresponding author.
E-mail addresses: u.usman.shaukat@ieee.org (U. Shaukat), imejaz@siswa.um.edu.my (E. Ahmed), zahid.anwar@seecs.edu.pk (Z. Anwar), f.xia@ieee.org (F. Xia).

http://dx.doi.org/10.1016/j.jnca.2015.11.009
1084-8045/© 2015 Elsevier Ltd. All rights reserved.

1. Introduction

In recent years, the increase in the use of mobile devices such as smart phones, PDAs, and tablets have morphed the computing landscape. Wide availability of Internet access through 3G/4G/LTE technologies and advances in portability and capabilities of mobile devices have given birth to a plethora of sophisticated mobile application frameworks. Mobile devices which were once dumb telephony devices have now become powerful enough to support a variety of day-to-day computing needs ranging from e-commerce, multimedia, gaming, to social media. These applications are often resource hungry and consume valuable mobile resources, such as battery and storage, much to a users' dissatisfaction. To overcome these issues, researchers have integrated mobile applications with services on the Cloud.

Cloud services allow on-demand access to computing resources, such as storage, processing power, and network access, with minimal setup delay and management effort. An application running on a mobile device offloads resource-intensive portions of the application to the Cloud, which performs the bulk of the computation and returns the results to the mobile devices, thus conserving mobile resources (Opera Browser, https://www.opera.com/; Dolphin, https://www.dolphin.com/; CloudMagic). Moreover, Cloud services allow sharing of resources between different devices, without actually having to manually synchronize the data. Noteworthy among these are Google docs (https://cloud.google.com/products/cloud-storage/) and Dropbox (https://www.dropbox.com/business/cloud-storage-and-backup). These Cloud services are designed in such a way so as to transparently support a large number of applications with minimal management effort.

The Cloud and its associated services is a very powerful paradigm for overcoming the limitations of mobile platforms and is therefore being widely deployed. However, Cloud services for mobile applications have their own inherent limitations. Satyanarayanan et al. (2009) suggest that WAN latency is a fundamental obstacle in leveraging Cloud services on the mobile platform. Mobile devices use WAN links (3G/EDGE/GPRS) to access Cloud services. These links exhibit low bandwidth, high latency and unreliable and intermittent connectivity. It can be argued that while the introduction of 4G and LTE will result in better network connectivity there will be minimal improvement in the overall user experience (Koukoumidis et al., 2011) because firstly increase in the throughput is more visible for bulk data transfer as individual mobile users send data in small chunks (Falaki et al., 2010). Secondly, the inherent issues in wireless radio links for example noise, signal loss, and radio link wake up delay will continue to degrade user experience for applications based on Cloud services (Wang and Dey, 2012). Thirdly, the trade-off between enhancing computing resources (processing power, RAM, disk capacity) and enhancing usability features (weight, size, heat dissipation, battery life) means mobile hardware will always remain relatively resource poor (Satyanarayanan et al., 2009). Enhancing usability features, such as lightweight hardware sets, longer battery life, and smaller form-factor are generally considered more critical than enhancing computing resources. Hence, full use of processing power and other features of mobile devices is not common. For these reasons, Satyanarayanan (2010) propose an alternative paradigm called Cloudlets.

A Cloudlet is self-managing, resource-rich system which has high speed access to the Internet and Cloud services. It consists of one or more networked machines with high speed internal connectivity and readily available computing resources to leverage. Cloudlets offer numerous advantages over the Cloud such as lower latency, higher bandwidth, offline availability, and cost-effectiveness to name a few. A number of Cloudlet frameworks are proposed in the literature. These frameworks enable the following functionalities and services: (a) offloading code and application to a Cloudlet for processing (Satyanarayanan et al., 2009; Soyata et al., 2012a,b; Ha et al., 2013; Angin et al., 2012), (b) providing data storage and cache services for data processing, storage and retrieval (Achanta et al., 2012; Qing et al., 2013; Koukoumidis et al., 2011; Verbelen et al., 2012) and (c) improving network QoS by providing control procedures and mechanisms to customize different network components (Fesehaye et al., 2012; Benson et al., 2011; Chen et al., 2013; Dey et al., 2013; Matias et al., 2011; Satyanarayanan, 2013).

The notion of Cloudlet is however still in its infancy. Core areas fundamental to Cloudlets, such as discovery and advertisement, security and trust establishment, application offloading and task partitioning, device-Cloudlet communication protocols, data storage and exchange formats, deployment models, services billing and monetization, and Cloud-Cloudlet interoperability, still require extensive standardization efforts for achieving a sufficient level of maturity from the research community.

This paper surveys the state-of-the-art Cloudlet frameworks and identifies challenges in their augmentation with mobile devices. We systematically classify the current knowledge by devising a taxonomy on the basis of key parameters related to Cloudlet architectures, deployment and application models. The critical aspects of the current Cloudlet frameworks in Mobile Cloud Computing are thoroughly investigated and a comparative analysis is reported for the benefit of a new user looking for deployment options available based on strengths and weaknesses of specific architectures. The contribution of the paper lies in the categorization of frameworks on the basis of a taxonomy, investigation of pros and cons for a variety of scenarios, and a comprehensive listing of possible deployment scenarios along with their requirements. We also dedicate a portion of the paper specifically to discuss challenges in deploying a Cloudlet in local wireless network environments. In addition we list open issues for the guidance of researchers in selecting appropriate domains for future research and obtaining ideas for further investigation.

Table 1
List of acronyms.

Symbol	Description
2G	Second Generation
3G	Third Generation
4G	Fourth Generation
AP	Access Point
APC	Armoured Personal Carrier
BS	Base Station
CACTSE	Cloudlet Aided Cooperative Terminal Service Environment
CC	Cloud Computing
CSA	Cloud Security Alliance
DSDV	Destination Sequenced Distance Vector
EC2	Elastic Cloud 2
EDGE	Enhanced Data Rate for GSM Evolution
EE	Execution Environment
ENISA	European Network and Information Security Agency
GPRS	General Packet Data Radio Service
IaaS	Infrastructure as a Service
ISP	Internet Service Provider
LAN	Local Area Network
MANET	Mobile Ad hoc Network
MCA	Mobile Computation Augmentation
MCC	Mobile Cloud Computing
MOCHA	Mobile Cloud Hybrid Architecture
PaaS	Platform as a Service
PDA	Personal Digital Assistant
QoS	Quality of Service
ROI	Return on Investments
SaaS	Software as a Service
SM	Service Manager
VANET	Vehicular Ad hoc Network
VM	Virtual Machine
WLAN	Wireless Local Area Network
Wi-Fi	Wireless Fidelity
WSN	Wireless Sensor Networks

The rest of the paper is organized as follows. Section 2 explains the fundamental concepts of Cloud computing and Mobile Cloud Computing. Section 3 presents a taxonomy of Cloudlet frameworks proposed in the literature and classifies them on the basis of their scope and architecture. Section 6 highlights the key requirements for achieving a successful deployment in wireless local area networks. Section 7 discusses the deployment challenges and the impact on the network. Finally, section 8 draws concluding remarks with future directions. Table 1 shows the list of acronyms used in the paper.

2. Background

This section puts this work in perspective by briefly summarizing the concept of Cloud computing and Mobile Cloud Computing. It then draws on some of these background concepts and motivates the need for Cloudlets and the new opportunities they provide.

2.1. Cloud computing

The National Institute of Standards and Technology defines Cloud computing as (http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf):

> "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction".

The Open Cloud Consortium defines it as:

> "The ability to scale and provision computing power dynamically in a cost-efficient way and the ability of the consumer (end user, organization, or it staff) to make the most of that power without having to manage the underlying complexity of the technology".

Cloud computing is a paradigm for large scale distributed computing. It has its roots in different technologies that precede it including grid computing (Foster et al., 2008), virtualization (http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_12-3/123_cloud1.html), and service oriented computing (Wei and Blake, 2010). The basic idea behind Cloud computing is to make computing, both location and device independent. This enables a computing task, or data to be available at anytime, anywhere, and on any device provided that the device can connect to a Cloud service. Cloud computing offers a number of benefits including scalability, lower IT infrastructure cost, higher availability, support for mobility, platform/device independence, disaster recovery, and environmental friendliness to name a few. Cloud computing enables a device to increase its capabilities by leasing resources and software services without requiring actual ownership of the same. A user or a device can access, utilize and release resources available on the Cloud and flexibly pay for whatever was actually utilized. It also enables a user to demand resources on demand at run time thereby supporting scalability.

Cloud computing offers three types of service models namely Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). In the case of IaaS, the Cloud provides processing, storage, memory, and other resources to the consumer. The user has no control over the underlying Cloud resources but he controls the system resources, OS, and installed applications. In PaaS, the Cloud allows a user to deploy applications however in this case he cannot control the underlying infrastructure or the operating system. In SaaS, the Cloud allows a user to access only a fixed set of applications and services which are deployed by the service provider and the user has limited control. These service models enable a user to utilize Cloud resources as according to his requirement without having to invest in deploying and managing resources on his own.

Cloud computing has three different deployment models namely public, private and hybrid Cloud. In a public Cloud, services are made available to the general public by a services provider where the business model involves selling subscriptions to Cloud resources. The user can lease Cloud services and resources using a "pay-as-you-go" model. In a private Cloud model, the Cloud is deployed by an organization and is accessible to a restricted group of people such as the employees. Private Cloud offers a limited set of services, catering only to the needs of a restricted group of users and its resources are not made available to the general public. A hybrid approach is a combination of public and private Clouds that are bound together. This allows data and application portability and certain services running on the private Cloud can access complementary services running on the public and vice versa. Common examples of public Clouds include Amazon EC2, GoogleApps, Windows Azure, and Rackspace. Amazon EC2 follows the IaaS model, by allowing users to deploy their own VMs on the Cloud. GoogleApp Engine allows user application deployment (PaaS). Similarly, services such as DropBox allow users to access specific storage services hosted on the Cloud (SaaS).

2.2. Mobile Cloud Computing

According to Liu et al. (2013) Mobile Cloud Computing (MCC) is defined as follows:

"MCC is a model for elastic augmentation of mobile device capabilities via ubiquitous wireless access to Cloud storage and

computing resources, with context-aware dynamic adaption to changes in the operating environment".

MCC is a fairly recent addition to the Cloud computing paradigm, which aims at augmenting resource-poor mobile devices with Cloud services. In the recent past, the hardware capability of mobile devices has enhanced considerably rivaling that of desktop PCs from half a decade ago. It is argued that if this trend continues, future mobile devices will have sufficient resources at hand for executing any workload. However, this will not be the case because the trade-off between enhancing computing resources (processing power, RAM, disk capacity) and enhancing usability features (weight, size, heat dissipation battery life) will always keep mobile hardware at a disadvantage in terms of resources. MCC therefore aims to allow a mobile device to access Cloud resources over a wireless link in a "pay-as-you-go" model. By using Cloud resources to perform various tasks, it promises to improve user experience by augmenting a mobile device so that it can overcome limitations faced due to shortage of resources such as computing power, storage capacity, and energy.

MCC is used to enhance the battery life of a mobile device by reducing power consumption during compute-intensive task execution (Kumar and Lu, 2011). This is done by offloading the compute-intensive task to the Cloud. It also improves the storage capacity of a mobile device by storing large chunks of data on online Cloud storage (Zhou and Huang, 2012). This technique is also effective in improving data reliability as Cloud services make use of multiple redundant storage servers. Mobile devices on the other hand are more prone to data loss due to theft and malfunctions. MCC also inherits the dynamic provisioning capability of the Cloud whereby mobile applications can be executed without having to reserve resources in advance. Similarly, it improves scalability by deploying applications in the Cloud where runtime reservation of resources enables them to meet flexible user demand (Kumar et al., 2013). Other benefits includes lower cost, support for a large variety of applications, and empowering mobile devices in deploying platform independent applications.

MCC follows the same service model as does Cloud computing. MCC model consists of three main components: (a) the mobile device, (b) the access technology, and (c) the Cloud service. A mobile device utilizes the available wireless signal (2G/3G/4G) to access the Cloud service and is usually bandwidth constrained, unreliable and suffers from latency issues. Therefore, applications with strict latency requirement cannot benefit from Cloud services. Mobile devices are forced to optimally utilize the Cloud services as best effort with minimum network communication (Ahmed et al., 2015a–c). Moreover, there is an extra cost incurred to the user of using a wireless link to access Cloud services. The latency, reliability, and cost involved in accessing Cloud services over radio wireless links overshadows the benefits of using Cloud services. To overcome these issues some solutions have been proposed including Ad hoc Cloud (Huerta-Canepa and Lee, 2010), PocketCloud (Koukoumidis et al., 2011) and Cloudlets (Satyanarayanan et al., 2009). Among these, Cloudlets are the most promising solution. Table 2 presents the comparative summary of Cloudlets and the Cloud.

Table 2
Cloud vs Cloudlet: a comparison.

Feature	Cloud	Cloudlets
Computing power	High	Intermittent
Resource elasticity	High	High
User experience	Satisfactory QoE	Excellent QoE
Availability	High	High
Client mobility	Support high mobility	Limited
Cost	High	Low-free
Latency	High	Low
Data safety	High	Intermittent
Management	Centralized – complex	Decentralized – self managed
Offline availability	Not available	Available
Bandwidth	Low	High
Storage capacity	Pay-as-you-go	High
WAN requirement	Persistent connection	None
State	Hard & soft state	Soft state
Network resource sharing	Large number of users (1000s)	Limited number of users (10s)
Deployment environment	Large data centers with controlled environment	Can be virtually deployed anywhere
Disaster impact	Catastrophic	Minimum impact

2.3. Cloudlets

According to Satyanarayanan, Cloudlets are a decentralized and widely dispersed Internet infrastructure whose compute cycles and storage resources can be leveraged by nearby mobile devices (Satyanarayanan et al., 2009). Abolfazli et al. (2013) define a Cloudlet as a proximate fixed Cloud consisting of one or several resource-rich, multi-core, Gigabit Ethernet connected computers aiming to augment neighboring mobile devices while minimizing the security risks, offloading distance and communication latency. Figure 1 depicts the general concept of a Cloudlet.

Unlike the Cloud, a Cloudlet exists at physical proximity to the mobile device, usually at a 1-hop distance and is accessible using a high speed wireless link such as Wi-Fi. Also referred to as a "data center in a box" (Satyanarayanan et al., 2009), a Cloudlet is a self-managing, resource-rich system, which can have high speed access to the actual Cloud. Architecturally speaking, a Cloudlet consists of one or multiple systems with high speed internal connectivity and readily available computing resources to leverage. In a Mobile Cloud Computing paradigm, a Cloudlet-based approach offers the following advantages over a cloud-based approach.

WAN latency: The strongest advantage a Cloudlet has over a Cloud is the improvement it offers in WAN latency (Verbelen et al., 2012). Since a Cloudlet is located at a one-hop distance it uses high speed WLAN links that do not suffer from latency issues. Many applications with real time latency requirements such as video streaming, real time image, speech processing, and interactive applications will benefit greatly from Cloudlets.

Higher bandwidth: The communication link between a mobile device and a Cloudlet is usually a WLAN (Wi-Fi) link. The bandwidth capacity of a WLAN link is typically two orders of magnitude higher than the bandwidth offered by wireless Internet links (EDGE/GPRS) (Satyanarayanan et al., 2009) used by mobile devices to access Cloud services on the Internet. Consequently an application which relies on bulk data transfer such as image processing, file transfer, and video editing/streaming will experience a huge difference. While an improvement is expected in mobile Internet bandwidth with the introduction of technologies such as 4G and LTE, the proportion of improvement in WLAN bandwidth is expected to be even larger.

Offline availability: Cloud services can only be provisioned if an active Internet connection is available. In the case of non-availability of an Internet connection, Cloud services are inaccessible. Cloudlets on the other hand are deployed on the local LAN or nearby 1-hop devices. The connection to the Cloudlet and its services do not depend on the availability of an active Internet connection. Mobile devices can therefore utilize Cloudlet services ubiquitously at any time.

Cost effectiveness: Accessing Cloud services incurs two types of expenses namely the cost of the Cloud service itself as well as the cost incurred for the Internet service used to access the Cloud (Aljabre, 2012). Cloud providers are commercial enterprises whose

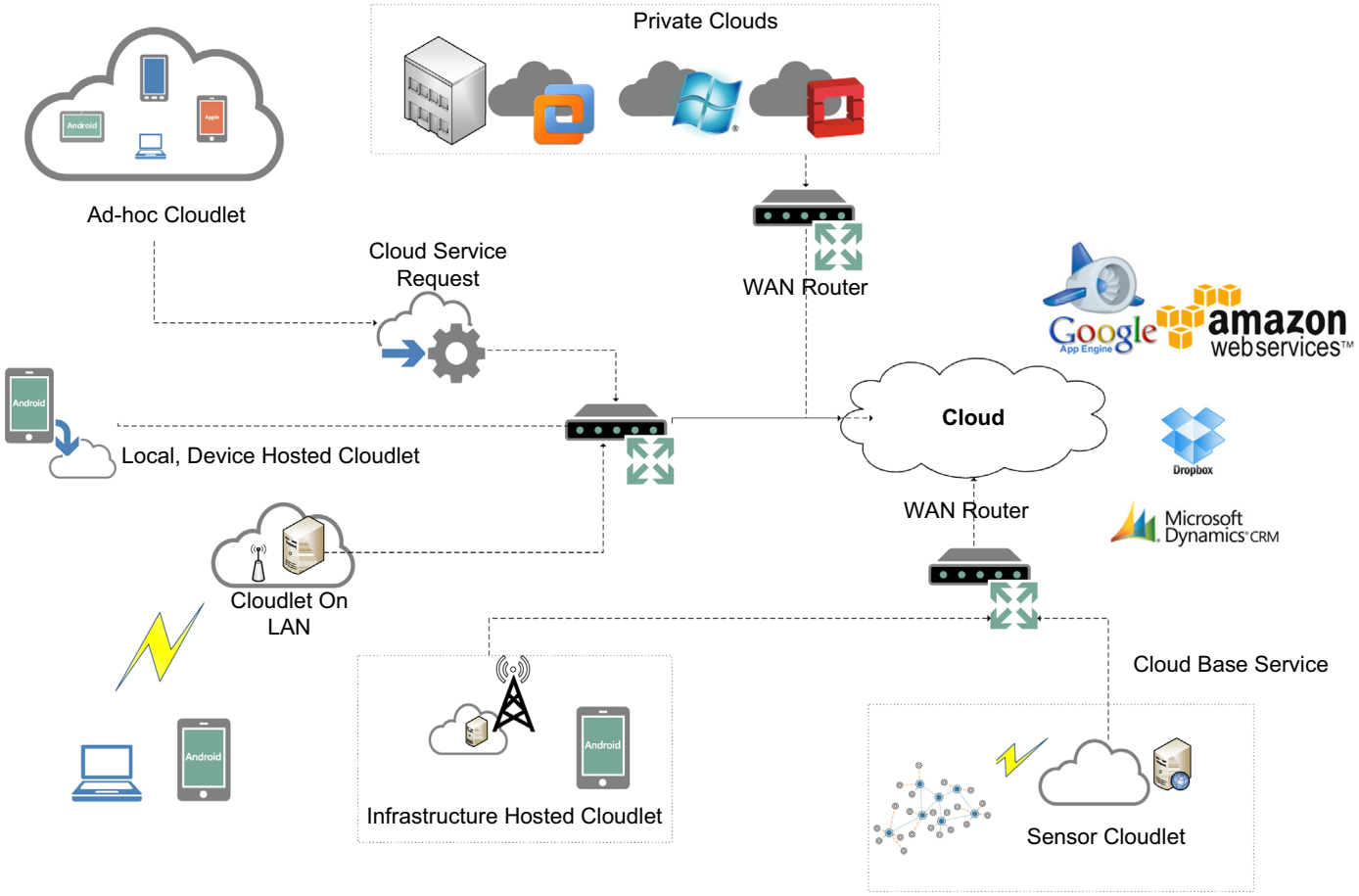

Fig. 1. Cloudlets can be deployed in a LAN environment, as a part of the infrastructure of communication services providers, in sensor networks, ad hoc Cloudlets formed by mobile devices or in a business environment by utilizing private Clouds.

business model relies heavily on profits gained from Cloud services. Therefore Cloud services today are more feasible for corporate clients, while individuals shy away if the cumulative cost exceeds a particular threshold. On the converse Cloudlets can be very cost effective in that a user does not need to pay for the Internet service as Cloudlets are available at one hop LAN/WLAN links (Lewis et al., 2014a). Cloudlets will be mostly deployed in small organizational entities similar to the deployment of Wi-Fi hotspots (Ahuja and Rolli, 2012). This ensures that Cloudlet services can be provided either free of cost (to attract clients to buy a special product) or provided at minimal price. Certain deployment models exist e.g. ad hoc Cloudlets where services will be available in return for extending one's own resources to the Cloudlet.

Disaster recovery: A Cloudlet, as opposed to a Cloud, maintains a minimal amount of data or soft state of a mobile application. The data transferred to the Cloudlet is either generated code or cached copy of data present on the mobile device for processing or storage (Koukoumidis et al., 2011). Loss of data or abnormal execution of application at the Cloudlet therefore causes minimal damage.

Management: The Cloud architecture supports a "pay-as-you-go" model which provides a seamless mobility experience for the user. However, the provider has to deal with a number of technical issues behind the scenes to manage a highly dynamic environment (Barbosa and Charão, 2012). Complex issues such as workload, capacity, and network management require extensive planning, implementation and administration effort. Cloudlets exhibit a similar and transparent user experience but unlike the Cloud, a Cloudlet provider provides a substantially simpler service set to a comparatively smaller number of users. This allows Cloudlet services to be self-managed, requiring minimal management effort (Satyanarayanan et al., 2009).

3. Cloudlet taxonomy based on augmentation models

Mobile Cloud applications are gaining tremendous popularity and Cloudlets promise all the benefits without being WAN-limited. As a mobile computation augmentation technique, a nearby Cloudlet allows a mobile device to overcome its resource poverty without relying on a distant Cloud. Abolfazli et al. (2013) have categorized applications which can benefit from resource augmentation into data-intensive, computation-intensive, and communication-intensive classes. Devices with computation-intensive applications are augmented by Cloudlets using computation offloading techniques. Devices with data-intensive tasks are augmented using data staging techniques. Finally devices with communication-intensive applications are augmented using Cloudlets which implement a network-as-a-service model. This high-level Cloudlet taxonomy is represented in Fig. 2 and each class of techniques is detailed below.

3.1. Computation offloading

Computation Offloading techniques aim to enhance the computing capabilities of a mobile device by migrating an application as a whole or by part from the mobile device to a proximate computing device. These techniques empower a mobile device to perform tasks which are beyond its computing capability. Computation offloading is not a new approach and belongs to a group of techniques referred to as Mobile Computation Augmentation

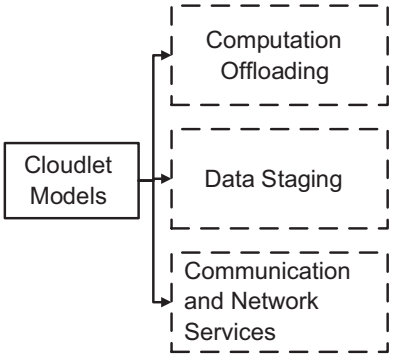

Fig. 2. Cloudlet taxonomy.

(MCA). Other approaches that can utilize resources in the Cloud or the Cloudlet to augment mobile devices include Load Sharing (Othman and Hailes, 1998), Remote Execution (Rudenko et al., 1998), and Cyber Foraging (Satyanarayanan, 2001). For applications based on the Cloud, offloading solutions rely on both Internet connectivity and Cloud resources as well as strategies that couple mobile devices with servers at deployment time. For mobile devices with unreliable WAN connections, these solutions can produce less than optimal results. On the converse in a Cloudlet environment there is a single hop proximity of the mobile device to the Cloudlet which alleviates some of these issues. The application can utilize offline availability to data (can cope without an active Internet connection), and can be easily partitioned either statically or dynamically into client and server portions to execute on the mobile device and Cloudlet respectively.

Based on the granularity of the offloading component, we can categorize computation offloading models into three main categories: (a) application offloading, (b) component offloading, and (c) VM image. The taxonomy of computation offloading techniques is given in Fig. 3 and are discussed briefly below.

(1) *Application offloading model*: In application offloading, the entire application or process is offloaded (Yang et al., 2015). It commonly encompasses Cloudlet discovery, context gathering, process migration, and remote execution control. It is the simplest model to implement as there is no need for program modification or partitioning for remote execution. There is no scheduling or task synchronization overhead involved. However, it does unnecessarily consume energy if the application and related data is bulky (Lago, 2013). Similarly application offloading is not feasible in scenarios where a constant input stream is required from a mobile device. If the connectivity with the Cloudlet is lost or process is terminated abnormally, the entire application needs to be re-executed from start. Application offloading techniques can be further categorized as follows.

Application virtualization: In application virtualization as illustrated in Fig. 4, the application is completely independent of the mobile device since the Cloudlet emulates the OS functionality. The Cloudlet run-time components intercept the application system calls and redirects them to the mobile device. The application itself remains unaware that it is interacting with a virtualized OS. In this technique (Lewis et al., 2014b), the device first sends the application package and meta-data to the Cloudlet. The Cloudlet then deploys the application and informs the client that the application server is ready for execution. The end user views and interacts with application over the network via a remote display protocol.

Cloudlet push: In the Cloudlet push model as illustrated in Fig. 5, a mobile device first discovers a Cloudlet and then sends an application along with its metadata to the Cloudlet (Lewis et al., 2014b). The Cloudlet deploys the server application on its

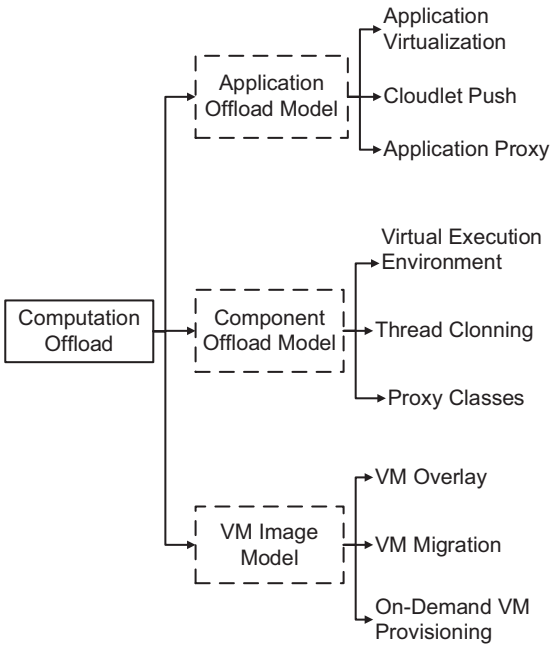

Fig. 3. Computation offloading taxonomy.

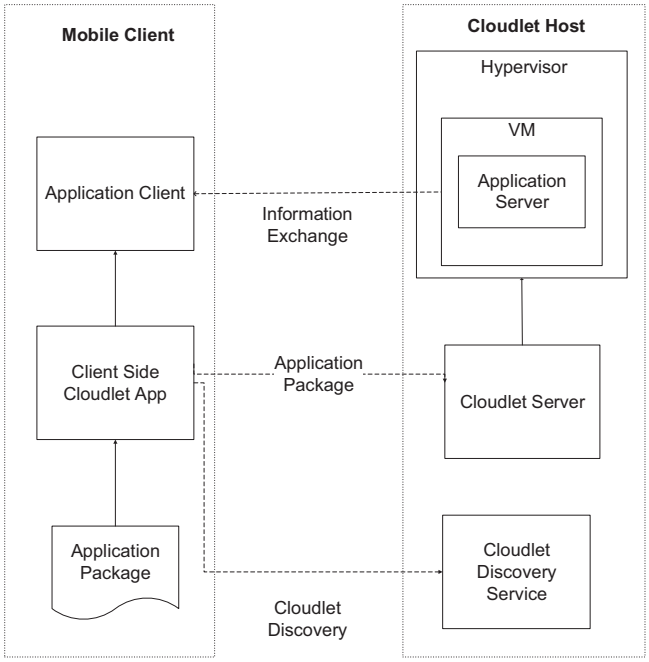

Fig. 4. Application virtualization.

platform, and pushes a client application to the mobile device. The client application is used to interact with the deployed application on the server. The mobile device then installs the client application and starts the client application.

Application proxy: In the application proxy model (Satyanarayanan et al., 2013), the actual application runs in the Cloudlet. A light-weight version of the application, called the client application, runs on the mobile device. The client application replicates every user action on the application running in the Cloudlet. The actual processing is performed by the application running in the Cloudlet and the final result is collected by the client application from the Cloudlet and displayed/stored on the mobile device.

(2) *Component offloading model*: In the component offloading model, only a component or a thread of an application is offloaded

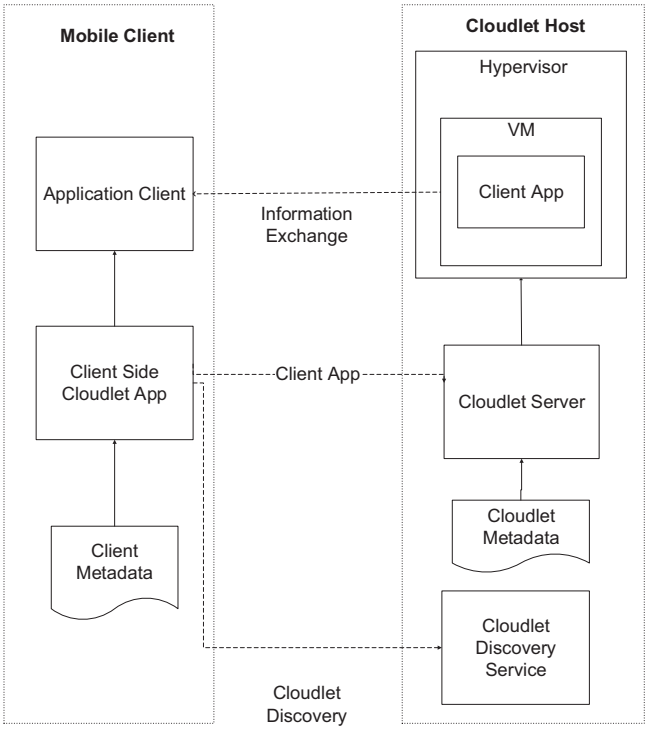

Fig. 5. Cloudlet push model.

to the Cloudlet (Ou et al., 2008). The offloaded component is normally the one performing the computationally heavy portion of the application. Such framework works by first partitioning the application into offloadable tasks (Liu et al., 2015). This partitioning can be done by the developer at design time (static partitioning), or can be performed by the system at run time (dynamic partitioning) (Cardellini et al., 2013). This strategy leads to large energy savings because methods that consume computational resources are offloaded. However, concurrent execution threads create data dependencies that are difficult to handle in an offloaded environment. Similarly scheduling and synchronization is required on the mobile device which adds additional overhead to the resource constrained mobile client. Overall Component offloading technique involves Cloudlet discovery, context gathering, partitioning, scheduling, thread migration, and remote execution control (Ou et al., 2008). Component offloading for Cloudlets can be further categorized as follows:

Virtual execution environment: Based on the concept of distributed shared memory, the heap, stack, and the locking states are kept consistent between the device and a virtualized environment created on the Cloudlet (Gao et al., 2012). This is usually achieved through an agent which synchronizes the states of the virtual heap, stacks, byte code sources, class initialization states, and synthesis classes between the device and the virtualized environment. Another part of the system, commonly known as a scheduler, performs the task of moving threads from one end-point to another in order to maximize the throughput (Verbelen et al., 2013).

Thread cloning: In thread cloning, an application is partitioned into multiple threads marked as either migratable or non-migratable thread (Chun et al., 2011). During the execution of a migratable thread, a system may decide to serialize the state of a thread and transfer it to a Cloudlet. On the Cloudlet, a clone of the original thread is created using the serialized state passed to it by the mobile device. This cloned thread takes over the execution from the original thread until the time it needs to return the control back to the original. Subsequently the Cloudlet saves the current state of the thread, serializes it, and passes it back to the mobile device and destroys the local clone.

Class offloading: Unlike other offloading mechanisms, where only objects currently being used by the thread are offloaded, in this technique, the class and all of its associated objects are offloaded to the Cloudlet (Hassan et al., 2014). Choosing classes to offload can be done by static analysis or during run time based on interaction between the migrating thread and a class. The advantage of this technique is that it is easier for a virtual machine to keep track of classes than objects and does not involve keeping track of creation and destruction of monitored objects (Shivarudrappa et al., 2011).

(3) *VM image model*: A Cloudlet architecture based on a virtual image model involves deployment of a VM on a Cloudlet infrastructure to provide resources to a mobile device (Satyanarayanan et al., 2009). This approach is independent of the architecture and software environment of the host Cloudlet and allows the Cloudlet provider to flexibly host as many VM guest environments as he wants, thus increasing the chances of a mobile user finding compatible Cloudlets anywhere in the world. VM based offloading models are classified as:

VM migration model: In the VM migration model, a running application is encapsulated in a VM image created on the mobile device. The VM is then saved, migrated to a Cloudlet (Jararweh et al., 2013), unpacked and then run, which in turn executes the encapsulated application. This process involves a complete VM image migration from the mobile device to the Cloudlet. After the application execution has finished, the VM state is saved and the image is transferred back to the mobile device. This involves application termination and suspending the VM instance in a normal manner. For successful VM migration, the mobile device must have an available VM image in a suspended and resumable state and the Cloudlet must have enough resources available to successfully execute the VM starting script (Yu et al., 2013).

VM overlay model: Also referred to as 'Dynamic VM Synthesis' or 'Transient Customization' of a Cloudlet, this idea was first put forward by Satyanarayanan et al. (2009) whereby a Cloudlet employs dynamic creation of a VM. In this approach, as illustrated in Fig. 6 a small VM overlay, much like a bootstrap configuration file, is delivered by the mobile device to a Cloudlet. The Cloudlet possesses the base VM from which this overlay was derived. The overlay is applied to the base VM and launches the modified VM. The VM then starts execution in the same state as defined in the overlay file. The components critical for successful VM synthesis include a VM customization script (VM overlay), a VM clean up script, and a base VM.

On-demand VM provisioning model: In this technique, as shown in Fig. 7, a Cloudlet creates an on demand VM to match the requirement of a mobile node (Echeverria et al., 2014). A mobile device, after discovering a Cloudlet, transfers a provisioning script to the Cloudlet. The Cloudlet then needs to compose a VM which fulfils the criteria set in the provisioning script. For this purpose the Cloudlet can instantiate an already configured VM from a VM repository that contains the full server components or it can take a base VM and install the required components (Wang et al., 2011). The difference between on-demand VM provisioning and VM synthesis is that in the latter the Cloudlet only needs to host the base VM while the overlay is provided by the mobile device. On the other hand, in on-demand provisioning there is low run-time transfer cost, but the Cloudlet needs to host the base VM, the server components and a provisioning software to be able to successfully fulfil a request.

(4) *State-of-the-art offloading techniques in Cloudlets*: Many frameworks, proposed in the literature, implement computation offloading techniques to augment mobile devices using Cloudlets. Application domains which can benefit from computation

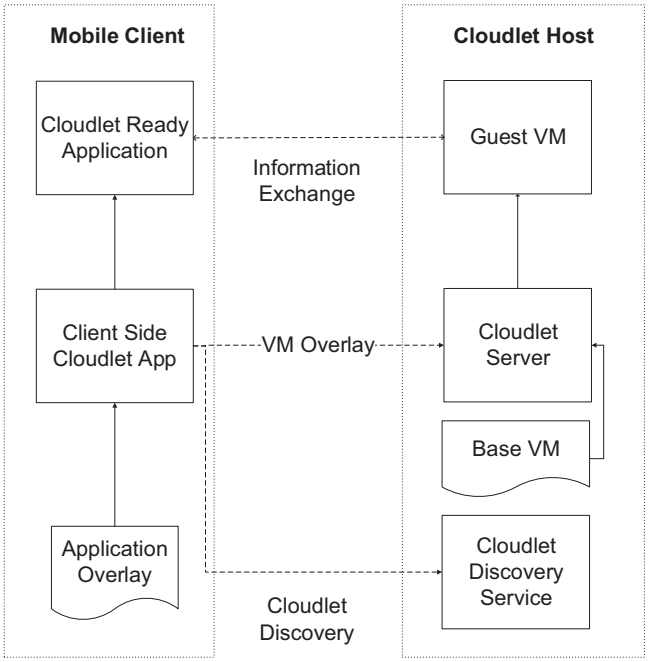

Fig. 6. VM overlay model as described in Satyanarayanan et al. (2009).

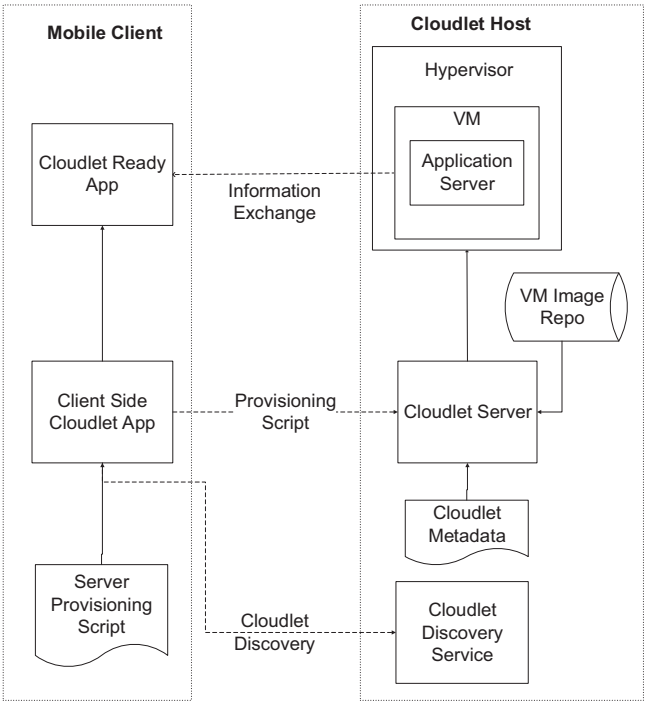

Fig. 7. On-demand VM provisioning as described in Satyanarayanan et al. (2009).

offloading to a Cloudlet include natural language processing, speech recognition, image processing, delay sensitive applications such as video and audio encoding and streaming, 3D gaming, and location-aware services. It is sometimes argued that current generation of smartphones and mobile devices have enough computing resources available to run such applications without the need to leverage the Cloud. Clemons et al. (2011) describes many computer vision applications that can run smoothly on mobile devices today. However, on the contrary, the computation requirement of these applications varies with respect to the operating environment and resources at hand.

Many studies have been performed to evaluate the performance of speech and face recognition algorithms in different context. Ha et al. (2013) performs experiments to evaluate response times of mobile applications and humans under different operational conditions. The study concluded that face recognition applications, under worst operational conditions, have a response time of 4.02 s as compared to 620 ms required for a human subject. This performance can be improved considerably if resources from an accessible resource-rich device are leveraged.

Many studies (Angin et al., 2012; Chandramouli et al., 2012; Dey et al., 2013) have shown that utilizing resources of a proximate device or node considerably improves the response time of delay sensitive real-time applications such as video streaming, 3D gaming and collaborative chat. Due to relatively smaller number of users on a Cloudlet, lower delay because of WLAN, and higher available bandwidth, utilizing Cloudlet resources produce better results than utilizing the Cloud resources. In the literature many frameworks exist which aim at improving the performance of applications by leveraging Cloudlet resources for computation offloading.

In a technique proposed by Satyanarayanan et al. (2009), a small VM overlay, much like a bootstrap configuration file, is delivered by a mobile device to the Cloudlet infrastructure. The Cloudlet possesses the base VM from which this overlay was derived. The overlay is applied to the base VM and launches the modified VM. The VM then starts execution in the same state as defined in the overlay file. In an image processing application, the VM could be a server that receives captured images from a mobile device and runs a face recognition algorithm against a set of saved images. Launching multiple VMs and installing image processing application through overlay results in parallel task execution using Cloudlets.

Soyata et al. (2012b) proposed a framework for application offloading to the Cloudlet referred to as MOCHA (MObile Cloud Hybrid Architecture) that uses a mobile-cloudlet-cloud architecture illustrated in Fig. 8. CloudVision (Soyata et al., 2012a) uses this framework to decrease the overall response time of a face detection and recognition task. The mobile device transfers the raw image (typically in 100s of KB) to the Cloudlet through a Wi-Fi or Bluetooth connection. The Cloudlet, which is capable of massively parallel processing and has a high speed Internet connection, dynamically partitions the face recognition task and distributes it among application instances running on Amazon EC2 instances on the Cloud. The partitioning is performed on the basis of processing and latency requirements of the application. The Cloudlet also acts as a buffer between the Cloud, where the actual program runs on virtual instances in parallel, and the mobile node. The Cloudlet saves the mobile node from transferring heavy

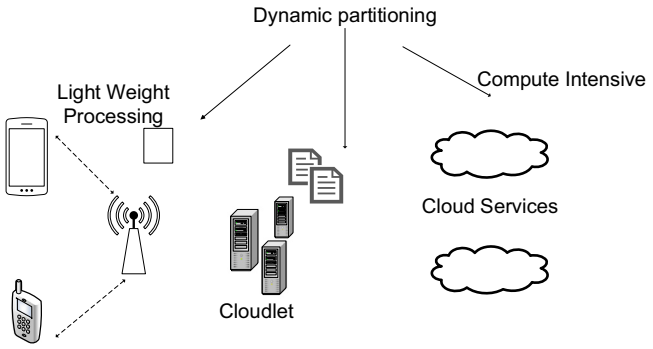

Fig. 8. MOCHA architecture as described in Soyata et al. (2012b).

images to the Cloud or perform dynamic partitioning of the task. In this framework, the bulk of processing is performed on the Amazon EC2 Cloud instance. This Cloudlet framework is therefore dependent on a high speed Internet connection to the Cloud. The partitioning is done before the execution of the task which means that any runtime changes in the latency requirement cannot be incorporated. The architecture of MOCHA provides little benefits if the problem space is small (recognizing faces from images with less than five faces and comparing them with a small number of other images).

MAUI (Cuervo et al., 2010) is a framework which uses the method offloading technique to offload computation to a Cloudlet. It works by converting the computation offloading problem to a graph partitioning problem so that one partition executes in the Cloudlet and the other executes locally. It uses .NET CLR to implement code portability. Before offloading, at initialization time, the framework profiles the device, application and the network to build a system linear energy model. It also profiles the CPU cycles required, execution time, current network conditions, and state transfer requirements. It converts the scenario to an energy optimization problem and solves it with the help of profiled information. MAUI uses managed code only and an application developed using unmanaged code cannot take benefit of the MAUI framework.

Fesehaye et al. (2012) present a Cloudlet framework design and analyze its impact on interactive mobile applications. The study focuses on interactive applications such as collaborative chat, video streaming and file editing. The study implements a customized DSDV protocol for inter-Cloudlet communication. The study concludes that applications on a Cloudlet always outperform applications on the Cloud framework provided the maximum number of Cloudlet hops are two or less. Cloudlets perform better by reducing data transfer delay and increasing throughput. In scenarios where the Cloudlet hops increase to four, the Cloudlet performance is inferior to that of a Cloud due to very high request transfer time. They propose to keep the number of hops to 2 or less by increasing the Cloudlet coverage area by using technologies like FlashLinq (Wu et al., 2013) or long range Wi-Fi (Chebrolu et al., 2006).

Another example of a framework which performs computation offloading at the component level is CloneCloud (Chun et al., 2011). Initially, a clone of the smartphone is created on the Cloudlet. The state of the primary phone and the clone is synchronized periodically or on-demand using an entity called the 'Replicator' in the CloneCloud architecture. The Controller, which invokes an augmented execution and merges its results back to the smartphone also interacts with the replicator to synchronize states while coordinating the augmentation. One of the main advantages of this technique is that it requires little or no modification to the existing application code. Unlike MAUI, it does not need the application to be written in a particular set of languages. The potential problem with CloneCloud is that it ignores the multithreaded nature of an application. Maintaining correctness of a multithreaded application is an important issue when multiple threads of execution depend on a shared state and some of the threads are migrated while some are not.

Zhang et al. propose a Markov decision process (MDP) based offloading algorithm for a mobile user in a Cloudlet system (Zhang et al., 2014). The algorithm divides the application into phases and the device can make a decision whether to execute each phase locally on the mobile device or offload to nearby Cloudlets. The phases can be either pre-determined or decided by the mobile user dynamically. It implements this decision problem as an MDP model to obtain an optimal policy to minimize the computation and communication costs. At the start of each phase, the user observes the local state as the number of jobs in the queue and the application phase the user is executing. Based on this state and the number of available Cloudlets it decides, on the basis of MDP, whether to offload the code. The main advantage of the technique is that it restrains from unnecessary offloading to Cloudlet. The offloading is performed only when the device processor has limited CPU cycles available. The potential problem with this technique is the dynamic partitioning into phases, which uses a guessing technique and does not produce optimal results.

Cardellini et al. (2013) propose a three tier architecture for computation offloading. This comprises a local tier of mobile nodes, a middle tier Cloudlet, and a remote tier of distant Cloud servers. It addresses the unmanaged scenario in which a user autonomously decides whether to offload to the nearby Cloudlet instead of the distant Cloud server. The Cloudlet in this framework offers a best effort service with no Quality of Service guarantees. It assumes that each user is selfish in nature and decides according to its own utility goals. The framework models the system as a queueing network and formulates the problem as Generalized Nash Equilibrium Problem. It takes power consumption and user experience as QoS measures to optimize the problem. A distributed algorithm is then run to solve the offloading decision problem for each user. This approach, while theoretically feasible, still has not been applied in a real environment.

In their work, Verbelen et al. (2012), propose a Cloudlet framework, which implements computation offloading at the component level. Each component is distributed within the Cloudlet to be executed independently. It consists of an Execution Environment (EE) which manages all the components and distributes them among the Cloudlets. If components depend upon each other, they can communicate through stubs and proxies, which in turn implement Remote Procedural Calls (RPCs). Each Cloudlet node hosts an EE and is managed by a Node Agent (NA) and the entire Cloudlet is managed by a Cloudlet Agent (CA). The CA has the capability of spawning new nodes if required. This architecture is similar to VM-based Cloudlets proposed by Satyanarayanan but with an extra middleware (EE and NA). The component is offloaded to a Cloudlet only if the resource available to the component does not meet the constraints as described by the application developer. The constraints and configuration of each component is defined in an XML file. One of the main disadvantages of this technique is that the components and their requirements are defined by the application developer and dynamic partitioning is not supported.

3.2. Data staging

In the recent past, Cloud storage services (http://www.infoworld.com/article/2871290/cloud-computing/understanding-cloud-storage-models.html) have changed the way we store and share data. Previously, data residing on one device was transferred to another by physically connecting the two or by transferring data using mass storage devices. Now, with readily available high speed Internet access and Cloud storage services like Google Drive (https://cloud.google.com/products/cloud-storage/), Dropbox (https://www.dropbox.com/business/cloud-storage-and-backup)) and Live Drive (http://www.livedrive.com/), users are more comfortable in keeping their data on the Cloud for ready access on a plethora of devices. The trend of sharing content over social media has added icing to the cake. Now people store their photos (Flickr, https://www.flickr.com/), videos (Vimeo, https://www.vimeo.com/) and other sharable documents on Cloud storage services, and share it with their friends, using one click sharing options. The need to synchronize mobile phone, laptops and PCs is more or less eliminated.

Cloud storage has also benefited business and enterprise users. Cloud enabled services such as web Cloud hosting, Cloud databases, and Cloud-based data backup storage are very popular among normal users, enterprises, and big data companies. Cloud storage not

only cuts cost by introducing a pay-as-you-go model, but also enables small and medium businesses to enjoy services, which were once, only affordable by large companies and enterprises.

Cloud storage provides huge benefits to diverse group of users, but it has its down-sides (Ji et al., 2012). Data services are bandwidth hungry in nature. Radio links, which are used by mobile devices to connect to Internet cannot satisfy this hunger. When they do, the cost is high enough to kill any advantage associated with these Cloud services (Abadi, 2009). Using existing cellular technologies, while it is feasible in issuing commands to copy data from one place on the Cloud to another, it is inefficient to transfer the actual data. MCC application developers struggle with the question whether to use Cloud storage services or go for other legacy solutions for data transfer and storage (Nelson et al., 2011). Cloudlets are now offering an apt solution to this very problem where clouds failed to do so. Cloudlets offer an always-on access to storage services over a high speed, high bandwidth Wi-Fi connection. Even if a Cloudlet has inadequate storage capability, it can always turn to a distant Cloud by using its own broadband connection (it is highly likely that one such connection is present).

The rise of modern social networks allows an unprecedented level of interaction among users. The increased interaction among applications working on the behalf of a user can cause problems. While users may be technically sound enough to utilize Cloud storage for data sharing, they may lose track of what applications have which information and what data is being transferred in the background without their knowledge. A Cloudlet can be introduced in such a setting, as a virtual representation of a user space on the Cloud (Qing et al., 2013). It can store user personal data and metadata regarding user applications providing a picture of which user data is being accessed by which application. For a user, a Cloudlet allows control of what information is accessible to Cloud services. For a developer, the Cloudlet provides a service API for the management of user Cloud storage and preferences thus acting as a local proxy for remote Cloud services.

Based on data storage model, we can categorize data staging techniques into four main categories: (a) file storage model, (b) data grid object model, (c) relational DB model and (d) virtual NAS model. The taxonomy of computation offloading techniques is given in Fig. 9 and the various categories are discussed briefly below.

(1) *File storage model*: File Storage is the simplest model to implement and deploy. This model relies on storing user files on the Cloudlet for storage or processing (https://www.dropbox.com/business/cloud-storage-and-backup). The model augments the mobile device by providing storage space to the device to store its files. The customer has access to either a block storage or a file system access through a file system protocol. In this model, the actual storage space is thin provisioned and the user is billed on the basis of actual usage (Devanathan Nandhagopal et al., 2012). Compression services can be used to reduce the actual space consumed. File storage model can further be characterized into models that use NFS/FTP protocols to store and retrieve user files and models that support storing, querying and retrieving semi-structured data.

NFS/FTP-based models (Liu, 2013) store user files on a file system and provide access to users over a network using NFS or FTP protocols. Data is stored on small virtual disks on the Cloudlet. These virtual disk can be mounted and unmounted on demand to allow a user to access the files. This type of model provides the simplest user interface, but lacks querying and indexing features.

Semi-structured data storage model stores data in semi-structured format (Ji et al., 2012). The data and its metadata is stored in the form of XML, JSON, Google protocol buffer or similar semi-structured data format. This model provides querying, indexing, and smart retrieval of data. The user files, however, are still stored on a file storage system that has inherit issues of data corruption, security, and consumption of large space due to absence of data compression.

(2) *Data object model*: A user application may want to access data from multiple resources. Cloudlet storage based on data grid model, enables searching, retrieving, and storing of data from numerous available data sets for the required data. Data Grid (http://www.gigaspaces.com/xap-in-memory-caching-scaling/datagrid) is a relatively old concept and was extensively researched during 1990s and early 2000s because it supports data-intensive applications. A Cloudlet providing data storage services can use this model to effectively provide service to data-intensive applications, which requires data from multiple data stores. While it is based on grid technology, it has the ability to support run-time provisioning of resources similar to a Cloud storage (Plattner, 2009).

Graph Object model (Shao et al., 2013) stores user data as objects, with each object having attributes to define itself and its properties. The user data, if in the form of a file, is stored and its reference is stored in the data object. This model is commonly used by applications, which supports the web standard such as SOAP and ReST. Each data object can be kept in memory or stored in a temporary cache for fast retrieval and querying. Storing data as graph objects also enables sorting and storing data in a hierarchical manner.

In-Memory, data grid object is a structured object that resides entirely in RAM (Plattner, 2009). A Cloudlet system with a large amount of RAM can support In-Memory data grid model to provide enhanced storage capability as read and write from RAM is much faster than from a hard drive. Also, storing users' data on a third party Cloudlet faces challenges such as perceived lack of control over the storage, security concerns, possible data loss, and fear of use of data without users' permission. In In-Memory data grid model, the data is not stored on the Cloudlet but is made available for processing in the Cloudlet layer via in-memory replication. Any changes made to the data are reflected back upon the on-device data store in near real-time.

(3) *Cloudlet DB model*: In this model, user data is stored and accessed from a database which is deployed in the Cloudlet (Abadi, 2009). The data is accessed directly from the database without using Cloudlet services or any Cloudlet specific API. Different deployment models exist for deploying a database in the Cloudlet (http://www.networkworld.com/article/2162274/cloud-computing/10-of). A user can either maintain a database independently in a Cloudlet on a VM image, or he can use a database-as-a-service whereby the database is managed by the Cloudlet owner and the user only pays for its usage (Deka, 2014).

A Cloudlet may provide a web-based console or an API to access, manage and maintain a user database on the Cloudlet. The database hides the underlying platform from the user application, thus providing platform independent data services. The

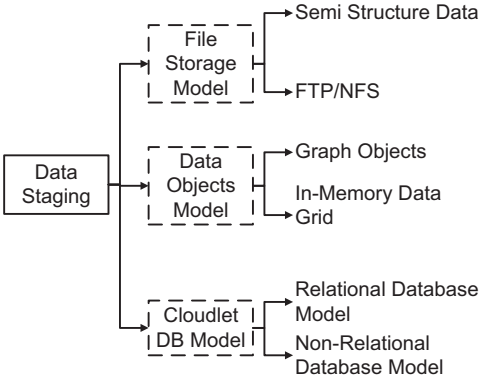

Fig. 9. Cloudlet organization taxonomy for data staging.

mechanism to access the data depends on the type of database and the database vendor.

Relational databases are one type of database capable of running on the Cloudlet. However, they are not suited for Cloudlet and runtime provisioning and are difficult to scale. Many Cloud-based SQL database services have been introduced to address this issue (Abadi, 2009).

Non-relational is another type of database which can run on the Cloudlet. These databases are designed to support large amount of data read/write and are therefore able to scale up and down easily. Examples include Cassandra and couch DB and are most suited for deployment in a Cloudlet. Most of the current applications are developed around a relational database model and enabling support for non-relational requires a complete overhaul of the source code (Deka, 2014).

(4) *State-of-the-art data staging techniques in Cloudlets*: Many frameworks, proposed in the literature, implement data staging techniques to augment mobile devices using Cloudlets. Applications which can benefit includes data sharing, offline storage, data replication services, 3D gaming applications, and data based decision making applications.

OPENi is a Cloudlet framework proposed for such a purpose. The OPENi Cloudlet storage framework (http://www.openi-ict.eu/objectives/) will provide users with a single location to store and manage their personal data. OPENi will consist of open source APIs. These APIs will enable MCC applications to connect to multiple Cloud-based services. A user will have a single location where he can view data shared by each Cloud based application, manage permissions for these applications and manage his personal space on the Cloud through OPENi Cloudlet.

Certain issues still remain to be addressed in order for these Cloudlets to come into force. The security and vulnerability analysis of OPENi and similar frameworks are yet to be analyzed. A major concern is the fact that such a Cloudlet framework represents a single point of failure (McCarthy et al., 2015). If the Cloudlet is compromised, it will jeopardize data security in the worst possible manner. Secondly, with the myriad of new Cloud storage applications being launched on a frequent basis, it will be impossible for OPENi and similar Cloudlets to provide services to them unless some standard protocol for communicating between the Cloudlet and Cloud storage applications is enforced. Standardization efforts for designing such protocols are currently almost nonexistent.

Cloudlets can be utilized to exchange data between the Cloud and the mobile device. In such a setup, a mobile device connects to a Cloudlet and requests data from a Cloud service (Qing et al., 2013). The Cloudlet downloads the data on behalf of the mobile device using its own broadband Internet connection and then transfers it to the mobile device over Wi-Fi. The Cloudlet can also provide indexing and caching capabilities, enabling faster access to the data for future requests. Consider the following scenarios:

In a classroom, the instructor uploads the lecture and related educational content to a storage space on the Cloud. He then asks the students to download it from the Cloud storage (Koukoumidis et al., 2011). Instead of having the entire class download the material from the Cloud using low bandwidth radio links, all the students can connect to a local PC acting as a Cloudlet. The Cloudlet downloads the lecture from the Cloud when it receives the first request and transfers it to requesting device through Wi-Fi. For each subsequent request the Cloudlet serves the file from its cache, saving bandwidth and download time.

In a battlefield (Satyanarayanan, 2013), a unit is ordered to move towards a target area as quickly as possible after intelligence reports of the presence of a high value target. The map and action plan will be sent to each soldier via their mobile devices in a few seconds. The control center located at a remote site uploads the map and mission plan to a secure Cloud storage and an alert is sent to each soldier to download the mission plan and maps. The soldiers instead of connecting to the Cloud service directly, connects to a Cloudlet installed on an APC or Humvee. The Cloudlet downloads the mission plan and maps using its high speed satellite link and transfers it to the requesting device. Each subsequent request is serviced using the cached copies.

At a social gathering, many people are capturing pictures and videos using their cell phones and want to share the same using social media accounts. They upload the files to a Cloudlet, which in-turn uploads them to a Cloud storage and passes back the URL of the uploaded content. The user can then share the URL on his social media account without having to upload the actual data. When another user present in the same vicinity wants to view or download this shared data, he will request it through the Cloudlet. The Cloudlet searches its local index and if found, serves the user with the local cached copy.

In CACTSE (Qing et al., 2013), a content distribution system for mobile terminals is proposed. CACTSE mobile terminals connect to each other directly with the help of a Service Manager (SM), forming a Cloudlet to access content via on-line Internet access as well as offline (local) access. The SM is a device which has enough resources to act as a mediator between mobile devices. The SM does not have a cache of its own but maintains an index of files. The mobile device requests a file from the SM. If the SM has an entry of the requested file in its index, it extracts the file entry and directs the requesting device to the file, which holds the requested data. The data can then be exchanged in a peer to peer manner. If the SM has no index entry for the requested file it downloads the file from the Internet. The CACTSE architecture is given in Fig. 10.

CACTSE works well as a limited content distribution system. Lack of cache services means that a peer-to-peer mechanism has to be implemented to support the file transfer between mobile devices. Support for node mobility and dynamic update of indexes at the SM when a node leaves or reenters is also absent. In a highly mobile environment, this can impose a very high overhead. Support and integration with existing Cloud storage services is also overlooked.

Zhao et al. (2012) proposed a cache service based framework to resolve the issue faced while downloading bulk data. The framework places a mobile device mirror (usually a VM) on an application server present in the telecom's infrastructure or BS. The server usually has a wired link to Cloud services. A synchronization module is present inside the mobile operating system and synchronizes every application execution, keyboard input and user interaction with the device mirror. The mobile device mirror is a VM maintained by the application server. When the user wants to download a file from the Cloud, the caching service present inside the application server will first download the file to a cache storage and send a “file-cached” notification to the user via the synchronization message. When the user decides to download the cached file (based upon its current battery and storage state), he can download it from the mirror server directly without establishing a connection with the Cloud server. Once the smart phone starts to download the file from the mirror server, its corresponding VM will also download the file from the cache server.

This framework relies on the resource hungry radio link. The cache service saves the user from downloading data from the Cloud storage and eliminates heavy Internet traffic cost. However, it does not improve the actual data transfer delay as the mobile device uses its radio link for actual file download. With user files cached on the application server, concerns regarding data security and privacy will arise. The Cloudlet utilizes the ever increasing mobile storage to cache data for ready access. Caching data on a user's mobile device can improve the performance of many

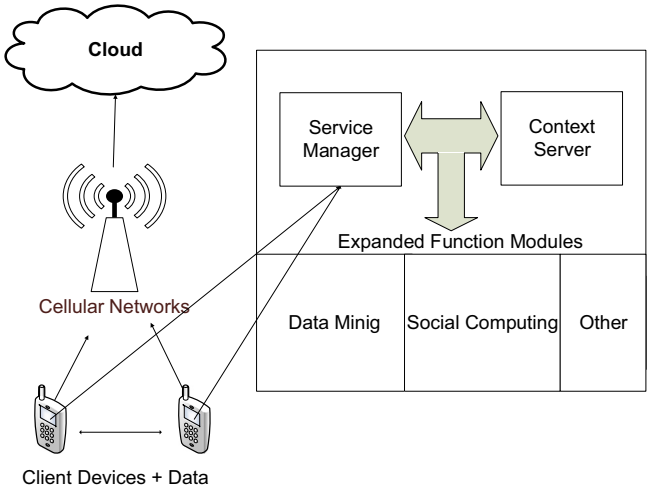

Fig. 10. CACTSE architecture as described in Qing et al. (2013).

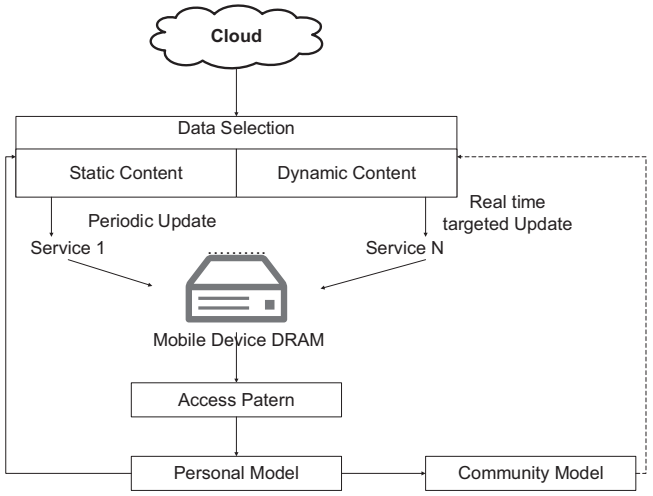

Fig. 11. Pocket Cloudlet architecture as described in Koukoumidis et al. (2011).

applications such as web searches, map and navigation applications and audio/video data for streaming.

Koukoumidis et al. (2011) proposed a Cloudlet framework known as Pocket Cloudlets. This Cloudlet service tries to predict which data may be downloaded by a user application in the near future, by analyzing access patterns of an individual user for a specific service (web search, audio and video data). The Cloudlet service then constructs a user behavior model. It also aggregates the behavior of multiple users into a community model. This helps the service to identify the most popular parts of the Cloud service data across all users. Both personal and community models are then used to identify the most frequently accessed parts of the Cloud service data. These parts are then downloaded in a cache storage. The Pocket Cloud architecture is explained in Fig. 11.

The downloading of data is performed only when Wi-Fi access is available and the mobile device is connected to an AC outlet. This ensures that the download process does not drain the battery resources. Later on, when a user tries to initiate a query or data transfer the Cloudlet service first searched its own index for required data and if found, serves it to the user. Luckily, studies have shown that the amount of dynamic data accessed by the user is small and dictated by personal interest and community trends. This means that if user behavior can be modeled accurately, there is a high chance of cache hit by a pocket Cloudlet.

Frameworks which propose caching of data on the user device face many challenges in determining exactly which data to cache, the update frequency and size and these have not yet been addressed completely by the state-of-the-art (Abadi, 2009). Many techniques try to balance between the amount of data to be cached and amount of storage available (Gordon and Lu, 2011). Modeling user behavior has its own limitation. Change of environment and community directly affects a user liking and interest. For instance, a user traveling or visiting a distant location will have different search interests than the user working in the office or studying at the library. Therefore, not only community but geographical location of the user needs to be considered while modeling a user's behavior. Although the storage available in a mobile device is increasing exponentially, the data on the Internet is growing at much greater pace. Therefore the storage available will always be less than the storage required. A determining factor is increasing the cache hit rate by the Cloudlet. Another serious issue is cache staleness. The larger the cache size, the harder it will be to keep the cache fresh and up-to-date. Therefore, any decision on the size of the cache needs to be made after considering cache refresh requirements (Sarwar et al., 2012).

Issues related to authentication, integrity, and confidentiality of data in the Cloud remain the biggest hindrance to its widespread adoption. These security issues in the context of the Cloud have their particularities, which can be attributed to openness and multi-tenant characteristics of the services. Many studies have analyzed popular Cloud storage such as DropBox, Microsoft SkyDrive and Google Drive services and have identified potential data integrity and leakage issues. One such analysis (Chu et al., 2013) shows that all three services have security weaknesses that may result in data leakage without users' awareness. Any Cloudlet storage framework will inherit some of these issues from the Cloud and has to address them before gaining wide spread acceptance.

3.3. Communication and network services

Ever since the advent of Cloud and utility computing, researchers have been developing ideas to shift network services from dedicated hardware to the Cloud (Bari et al., 2013). Network service providers and ISPs have invested in technologies, which utilize the Cloud and the Cloudlet to provide services such as Network as a Service (NaaS) Costa et al. (2012) to the end user. Services such as virtual private networks, bandwidth on-demand, mobile network virtualization and in-network services are now offered by ISPs using Cloud services. Cloudlets have the capability of providing the same services without the additional WAN delay. These services can be categorized as shown in Fig. 12.

(1) *Network service model*: In this model, a Cloudlet is deployed to provide different network services (Chen et al., 2013). In addition to regular data and voice services many ISPs provide content hosted on ISPs' servers, firewalls and content filtering, traffic optimizers, virtual private networks, and mobility support. A Cloudlet, due to its proximity to user devices and network infrastructure, can effectively provide these services with minimum cost and maximum benefits. In this model, the Cloudlet resides within the telecom infrastructure premises and can elastically distribute its resources among services hosted on it (Costa et al., 2012). A Cloudlet in such a setting can also act as a middlebox, providing services such as firewall and content filtering. Network service model can further be categorized into two types: (a) Quality of Service (QoS) provisioning model and (b) in-network content provisioning model.

The Quality of Service (QoS) provisioning model (Malathy et al., 2013) allows a Cloudlet to provide services to improve quality of service of the end user. In this model, suitable points of the

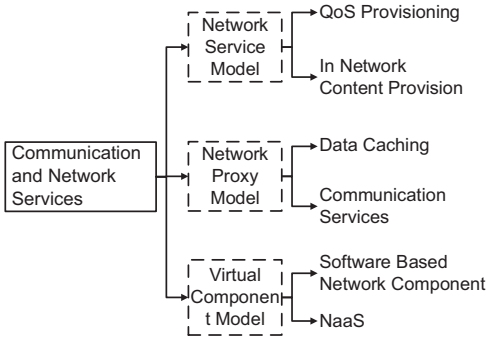

Fig. 12. Cloudlet organization taxonomy for network/communication service model.

presence are identified by the ISP for the deployment of the Cloudlet. The deployed Cloudlets can then act as smart devices, monitoring network traffic, and negotiating QoS requirements with the end user. They then allocate network resources to provide the desired QoS to the end user.

The in-network content provisioning model allows adding services in the network core using Cloudlets (Sherry et al., 2012). One of the main shortcoming of the current Internet is the fact that it is difficult to add new functionality and services to the network core. Many researchers have turned to overlay networks to implement value-added services. Cloudlets allow these services to be deployed within the core network without the use of application layer overlays. These Cloudlets act as service modules, deployed within the telecom infrastructure, offering services at the network level. Services can be added and removed at runtime using these Cloudlets.

(2) *Network proxy model*: In the network proxy model, a Cloudlet can act as a proxy to an Internet connection or to a Cloud service (Khalaj et al., 2012). This is helpful when a direct connection to the Internet or Cloud service is not available or direct communication to the Internet puts extra load on the already depleted resources of the mobile device. The Cloudlet can also host other services such as intermediate data processing, content caching and offline content storage. The Network proxy model can be categorized as follows:

A *Data Caching Cloudlet* caches intermediate data and serves it either on demand or after performing certain processing. Data caching also helps to save communication bandwidth by serving the same data to multiple devices. Many Cloudlets can prefetch data by modeling the users' behavior. For example a Cloudlet can predict user behavior based on past search pattern, user location, and other contextual information to cache and download prospective user queries. Such Cloudlets, if deployed on local device, provide the most benefits.

The *Communication service model* allows a Cloudlet to act as an Internet hot spot and provide communication services to connected devices. This allows a mobile device to conserve its energy by switching off resource hungry cellular signals and shift to Wi-Fi radio.

(3) *Virtual component model*: Cloudlets and Cloud-based networking will be a key ingredient of software defined networking paradigm in the near future. With software defined networking, certain aspects of management and control can be shifted to a Cloudlet, while switching and routing services remain in the hardware (Bozakov, 2011). Technologies such as OpenFlow (Matias et al., 2011) are allowing the dynamic provisioning of network resources by moving the decision making algorithms from hardware to the Cloud. Similarly, virtual routers and switches have now emerged, which exist within a hypervisor. The Virtual component model can be categorized as: (a) network-as-a-service model and (b) software-based network component model.

Network as a service model follows the same subscription and delivery model as software-as-a-service model (Costa et al., 2012). It moves all core networking functions, including addressing and the route calculation, into the Cloudlet and eliminates the need for any hardware other than that which provides an Internet connection.

Software based network component model involves Cloudlets, which act as network routers and switches. This is an essential part of Cloudlet setup based on virtual machines (Manzalini et al., 2014). Virtual machines do not need to have direct access to hardware router. Instead a virtual router can route network traffic between virtual machines and between a virtual machine and the outside world.

With the popularity of the Internet-of-things (IoT), the technology to connect multiple devices together to form an ad hoc or persistent network is now quite mature (Satyanarayanan et al., 2015). Internet of Vehicles, Internet of Sensors and Internet of Mobile Nodes are becoming ubiquitous. These and related ideas are leading the world to 5G networks (Bassi et al.), which aim to utilize the potential offered by hyper connected mobile devices. The research in this area is currently focusing on the dimensions of including fast creation and deployment of new and hidden services, service access models to handle diversity in access speeds and connectivity, and more importantly on devising mechanisms to utilize these services in an energy efficient and secure manner.

(4) *State-of-the-art in Cloudlet communication service model*: Cloud computing is moving beyond IaaS, PaaS, and SaaS, to Data-as-a-Service (DaaS) (Rajesh et al., 2012; Terzo et al., 2013), Knowledge-as-a-Service (KaaS) (Grolinger et al., 2013; Xu and Zhang, 2005) and Network-as-a-Service (NaaS) (Costa et al., 2012). Network-as-a-service in the Cloud (Benson et al., 2011) introduces the concept of NaaS, a framework in which Cloud service users have secure access to network infrastructure. In common practice, a Cloud user can select the computational and storage resources needed for his applications. However, he has no control over the network component. If a modification is needed in the protocol or forwarding logic, a user has to resort to implement these in unwieldy application layer overlays (Bari et al., 2013). These overlays are quite inefficient as compared to modifications made at the network layer. NaaS provides a framework, which allows the user to access resources such as routers and switches and performs modifications and customization as required by his application. Recently, the interest in programmable routers and software defined networking has increased and OpenFlow (Vaughan-Nichols, 2011) enabled network devices are looking to replace the current generation of routers and switches. It is now possible for Cloud users to perform forwarding decisions, do load balancing, manipulate packets in transit and reserve/allocate resources to provide quality of service guarantees required by certain Cloud applications. Services such as data streaming, complex event processing, data aggregation, content based networking and content centric networking can benefit greatly from programmable network resources in the Cloud (Verbelen et al., 2012).

Cloudlets with their inherent advantages of low latency and high speed connections to both mobile devices and to the Cloud, have obvious reasons to provide Network-as-a-Service support to Cloud users and the Cloud itself. In Fesehaye et al. (2012), the authors implement a centralized routing and signaling approach similar to OpenFlow. Instead of each node calculating shortest routes between itself and other nodes, a central server performs this task. The central server is located close to the nodes making up the Cloudlets. This ensures lower delays in route propagation. Deploying a routing server also enables centralized routing policy implementation and allows a single point to update and distribute new routes. Although the said framework implements the routing

node as a server within a Cloudlet environment, the same service can be provided by deploying a Cloudlet within a Cloud which exists close to the EDGE server and user nodes (Lewis et al., 2014b). The Cloudlet, which is a trusted entity deployed at a user premises (e.g. DSL modem) can expose methods to services running on user devices to program network infrastructure and deploy its own modification to current routing protocols.

4. Cloudlet taxonomy based on Cloudlet service model

The capability of a Cloudlet to augment MCC varies from framework to framework. Based on the service model, Cloudlets can augment a mobile device by offering services for:

- Processing Data.
- Caching Data.
- Partitioning/scheduling jobs.

Processing Data: Mobile devices are often limited in resources. Modern day mobile devices are equipped with multi-core processors and large banks of memory but fully utilizing these resources puts strain on another very rare resource i.e. the battery power. There are many applications that try to overcome this problem by utilizing resources on the Cloud. These applications need to transfer raw data to the Cloud using radio WAN links. While Internet transmission over radio links puts strain on the battery, the cost of using Internet over radio links is relatively high. By introducing a Cloudlet, a mobile device can offload the processing. The Cloudlet decreases the processing time by utilizing high speed processors and then transfers the results to the mobile device over low energy consuming WLAN link. A mobile device can decide to utilize a Cloudlet service when it needs to save battery energy, process large chunks of data with little delay, or have depreciated or low resources to perform the tasks at hand. For a framework providing processing resources, it needs to host a user application or a part of it to perform required processing on the input data. For such Cloudlets, IaaS and PaaS models are the most suitable.

Caching Data: A Cloudlet can utilize its internal storage to cache and store data and make it available to a mobile device at a later stage or when requested. A Cloudlet framework providing such services can have intelligent prediction and behavior modeling algorithms which can predict user queries based on past experience, location, and environment. Similarly, a Cloudlet providing these services can act as an intermediate proxy/cache server between mobile node and actual Cloud. It can cache data requested by one device in order to serve another which can request the same data in the future. Many applications can benefit greatly from these Cloudlets. For example, location-based services, such as nearby place finder and location-based social networking applications like Foursquare (http://foursquare.com) will show relevant data (hotels, shopping places, tourist attractions) close to a user's geographical location. A Cloudlet can preemptively cache data for such services based on location data and serve the cached data to any connected device. Consider the following scenario. A Cloudlet, which is a fixed PC, can download and cache GoogleMap files to its internal storage. When a mobile device connects and accesses GoogleMaps, the chances are, that it will request maps of nearby a location for navigation or place finding. The Cloudlet can then make the files readily available to the application.

Unlike Cloudlets providing processing services, a Cloudlet providing cache services does not need to host user applications in whole or by part. However, architectures of these Cloudlets need to follow specific data formats and data transfer protocols. For such Cloudlets, most commonly employed service model is SaaS.

Partitioning and scheduling: In the current mobile Cloud computing architecture, many mobile applications need to offload computation to the Cloud. If multiple Cloud servers, with different network and workload conditions, are available, the mobile device chooses either the first server or one that can offer the best possible service (low delay, better processing power, lower cost). In a geographically close environment, such as a classroom or a bar, several mobile devices can send their request to the same server but leave other servers idle. A Cloudlet in such an environment can act as a coordinator between the mobile devices and the Cloud server. It can keep track of the requests, current status of the requested server and current state of the mobile device. Based on this information a Cloudlet can inform the mobile device which server to connect to, to obtain optimum performance. The architecture for Cloudlet with such a service model will have no need to host the user application.

However, it needs to host an application capable of collecting information from devices and servers and perform all the processing required to coordinate among the devices. The most common service model which can be applied by these servers is SaaS.

Many MCC applications can partition a task into subtasks and send them to servers on the Cloud. For real-time applications, such as face recognition, speech synthesis, and virtual reality applications, actions such as capturing input, task partitioning, and scheduling them to multiple servers in real-time is not computationally feasible for a mobile device. A device can utilize a Cloudlet in this scenario by delegating the partitioning and scheduling job to it. Several scheduling strategies can be applied by the Cloudlet to increase the overall throughput and reduce the latency of the task. Partitioning of a task requires knowledge of internal structure of the task. This knowledge can be provided by the device to the Cloudlet as a meta-data, or a Cloudlet can host the actual user application and partitioning is done in the Cloudlet by the hosted application. It is highly unlikely, because hosting each application on a Cloudlet just for partitioning the task without doing any processing is not feasible. Architectures for such Cloudlets will have their own partitioning and scheduling services. The service model for such an architecture can be SaaS or PaaS. The Cloudlet taxonomy with respect to service model is shown in Fig. 13.

5. Cloudlet taxonomy based on architecture

The concept of Cloudlet is of a resource-rich device proximate to the mobile devices and connected via low latency local area networks, capable of lending resources to the mobile device to perform resource-intensive tasks. Cloudlets have the potential to address the performance and energy issues of MCC. Unlike the Cloud, where different architectures are already somewhat mature and deployed commercially, the concept of a Cloudlet is still in its infancy stages. Therefore most of the Cloudlet architectures found in the literature have not passed the test of actual deployment. Similar to the case of the Cloud, where Cloud architecture depends upon its service model (PaaS, SaaS, IaaS), Cloudlet architectures are also developed under the influence of its service model. Figure 1 demonstrates the concept of a Cloudlet deployed in a LAN environment. On the basis of architecture, Cloudlets can be categorized as shown in Fig. 14.

5.1. VM-based Cloudlets

A Cloudlet architecture based on virtualization involves deployment of a VM on a Cloudlet infrastructure to provide resources to a mobile device to execute tasks. A VM-based approach separates the VM environment from the Cloudlet infrastructure's permanent host

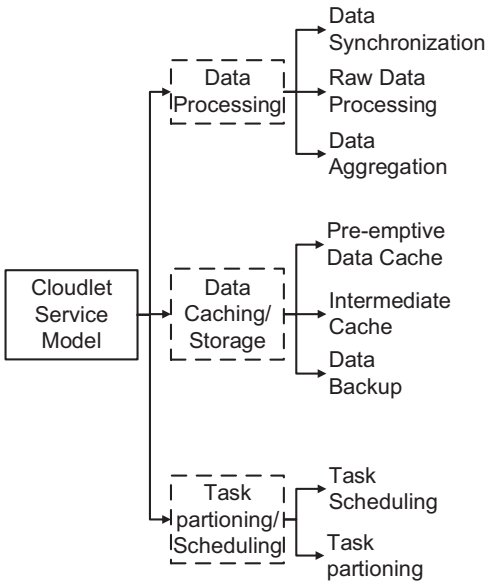

Fig. 13. Cloudlet taxonomy with respect to service model.

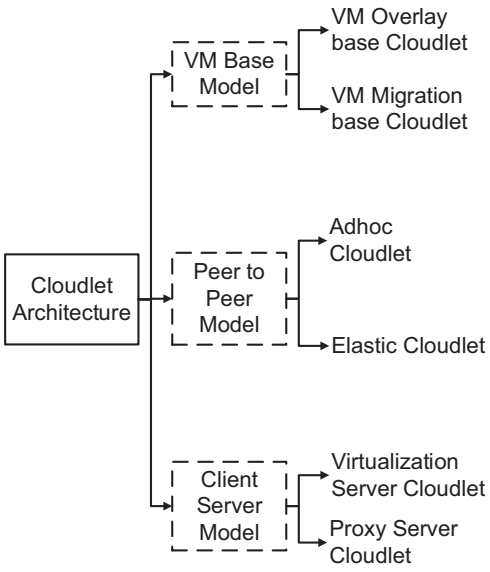

Fig. 14. Cloudlet taxonomy with respect to architecture.

software environment. This approach is independent of the architecture and software environment of the host Cloudlet. It allows the Cloudlet provider to host several VM guest environments and thus increases the chances of a mobile user finding compatible Cloudlets anywhere in the world. Two most widely used VM-based architectures are VM overlay and VM migration.

VM Overlay based Cloudlet: In this approach, as illustrated in Fig. 6 a small VM overlay, much like a bootstrap configuration file, is delivered by the mobile device to a Cloudlet. The Cloudlet possesses the base VM from which this overlay was derived. The overlay is applied to the base VM and launches the modified VM. The VM then starts execution in the same state as defined in the overlay file. The components critical for successful VM synthesis include a VM customization script (VM overlay), a VM clean up script, and a base VM. According to Satyanarayanan (Satyanarayanan et al., 2009), VM synthesis has an approximate delay of 60s to 90s which may not be acceptable for delay-sensitive applications if not intolerable for performing simple or ad hoc tasks on the Cloudlet. Dynamic VM synthesis also assumes that a small set of base VMs are sufficient to provide services to a large set of users. This assumption does not hold for VM-based on relatively older or newer versions of the operating system and therefore such VMs will not find a compatible Cloudlet.

VM Migration based Cloudlet: In the VM migration model, a running application is encapsulated in a VM image created on the mobile device. The VM is then saved, migrated to a Cloudlet (Jararweh et al., 2013), unpacked and then run, which in turn executes the encapsulated application. This process involves a complete VM image migration from the mobile device to the Cloudlet. VM migration technique can suffer from high transfer delays due to large VM size. Moreover, as the Cloudlet will be unaware of migrating VMs in a complete compiled form instead of an overlay, it can fail to allocate enough resources to service the VM. Establishing a trust relationship between Cloudlet provider and the mobile device is also an issue to be addressed properly. Although applications are executed within the VM, the data transferred between the mobile device and the VM will always pass through the guest operating system layer and is thus accessible to any malicious code present in the Cloudlet. A simple solution is to encrypt the data which is exchanged between the mobile device and the VM, but additional cryptographic operations add extra processing load both on the VM and mobile device and can also result in delayed results being received from the Cloudlet.

5.2. Peer-to-peer model

In the peer-to-peer model a mobile device utilizes the resources of the Cloudlet in a peer to peer fashion. In a traditional Cloudlet architecture such as VM-based Cloudlets and Client server model, the dependence is on the network operator to deploy a Cloudlet infrastructure. In the peer-to-peer model, individual devices cooperate in the Cloudlets. For example, different mobile devices in a home can cooperate with each other to form a Cloudlet. Similarly, mobile devices within close proximity such as trains and classrooms can negotiate and form a Cloudlet. The main advantage of this technique is that the dependence on the network operator to deploy a Cloudlet is alleviated. Another advantage of Cloudlet based on the peer-to-peer model is that instead of executing the entire application on a remote VM as a VM-based Cloudlet, the application can be partitioned into components and different components are then offloaded to various nodes in the Cloudlet. In this way, a more flexible means of resource allocation is possible in which components can be distributed among Cloudlet nodes according to the application requirement. That is, the latency critical part can be processed at the node while non-critical components can be offloaded to Cloudlet nodes. The peer-to-peer model relies heavily on discovering nodes in the proximity, which are ready to take part in forming a Cloudlet. This dynamic discovery of mobile nodes is critical in forming a Cloudlet. The main challenge is to deal with the heterogeneity of the mobile devices. Different types of mobile devices such as smart phones, PDAs, smart appliances and smart wearable devices have to support and agree on a single protocol to exchange and process data. Two widely used peer-to-peer Cloudlet architectures are ad hoc Cloudlets and elastic Cloudlets.

Ad hoc Cloudlets: In ad hoc Cloudlets, multiple nodes that are in physical proximity to each other form an ad hoc Cloudlet. Much like the creation of an ad hoc network, formation of ad hoc Cloudlet involves discovering physically proximate nodes, advertising available resources and exchanging data and components among themselves. In an ad hoc Cloudlet, the Cloudlet management can be central, i.e. one node can be selected as a master node, which performs all the management operations. It contains a global overview of the resources, current status of running

components and controls the distribution of components among the available nodes. Similarly, the management can be distributed in which each node keeps track of its own resources and periodically notifies all the other nodes in the Cloudlet of its available resources and current status of the executing jobs. The job distribution in this case is performed by the mobile device on the basis of the information it receives from each node. Verbelen et al. (2012) proposed a centralized ad hoc Cloudlet in which each participating node has a node agent, which communicates with a Cloud agent. Cloud agent is a node which manages the Cloudlet. Node agents can communicate directly with each other as well.

The running components on each node is managed by an EE. The EE manages and monitors the running components, and in case of any issues or resource exhaustion, notifies the node agent. The node agent in turn notifies the Cloud agent. The Cloud agent, which contains a global overview of the Cloudlet, calculates a new node to migrate the component from the troubled node. Hierarchical centralized approach of the architecture has advantages over a decentralized approach. Each node agent has to maintain a minimal amount of information. The complex decision making logic also resides on the Cloud agent and not on the weak nodes, thus allowing them to conserve their battery and computing resources.

Elastic Cloudlets: In elastic Cloudlets, peer mobile devices run in a virtualized environment. The environment consists of a Cloud agent, which has the capability of spawning new virtualized nodes, on demand, and close them when not needed. These nodes are spawned with node agent and execution environment already embedded in it. This type of Cloudlet is similar to VM-based Cloudlets. However, the main difference is that the node agent and execution environment are already embedded, and it only needs to support component offloading. Unlike a VM-based Cloudlet, complete application migration need not be supported. This type of Cloudlet can be deployed by the network operators and ISPs to support a peer-to-peer model within their networks. A Cloudlet such as this can be created on a virtualized hardware close to existing networking infrastructure (APs, Base Station, Private Cloud) of an already deployed network to support mobile devices interested in peer-to-peer Cloudlets only.

5.3. Client server model

Cloudlets using the client server model fall in the category of SaaS, where a particular service is available for mobile device to use. Most of these Cloudlet architectures are stripped down versions of an already deployed Cloud service. In this model, an application on the mobile device, instead of using its resources to carry out the task, passes the relevant input data and task details to the service running on the Cloudlet. The service in the Cloudlet performs the required processing, finishes the task and passes the output back to the mobile device. This model serves the same purpose as that of Cloud services running on the Internet but with an added advantage of negligible network delay and at a fraction of the cost. Following client server models for Cloudlets are found in the literature:

Virtualization server model: In virtualization server model, a Cloudlet provides mobile devices with the ability to launch VM images on its hardware resources. Unlike VM overlay or VM migration model, a Virtualization server model does not allow VMs to migrate from a mobile device to the Cloudlet nor does it provide any dynamic VM synthesis. The model allows mobile devices to run an available instance of a VM or virtual image of an application on its hardware, execute tasks and release the resources by saving the VM state and shutting it down. Common examples of applications that benefit from this model are the applications, which uses the worker thread approach. These applications, under stressful conditions, create worker threads of themselves on another machine and utilize it for sharing some of the load. For example, an application delivering content (Web Server, Temporary storage servers, Internet Proxy) may start a worker thread of itself on a virtualization server during heavy load, redirect some of the queries to the worker thread, and when load wears out, shutdown the worker thread without having to migrate data or an overlay configuration file.

Proxy server Cloudlet: In the Cloud proxy model, the Cloudlet works as a proxy to Cloud services. The Cloudlet is connected to the Cloud using a high speed broadband connection. The mobile device instead of connecting to Cloud connects to the Cloudlet and requests for the Cloud service through the Cloudlet. The Cloudlet either acts as a forwarder and forwards the received request to the Cloud on the behalf of the mobile device or acts as a mediator. In the case of a mediator, the Cloudlet does not forward the request directly to the Cloud but performs one of the following actions. It either serves the request using its own resources, or refuses the request based on some policy constraints.

6. Cloudlet deployment requirements

In this section, we have highlighted the key requirements that needed to be fulfilled for the successful deployment of the Cloudlets in WLAN. The highlighted requirements are straightforward guidelines for Cloudlet architecture designers, service providers, and prospective Cloudlet clients that can assist them in attaining a successful deployment of Cloudlets in WLANs. The key requirements are discussed below.

6.1. Investigating Cloudlet deployment incentives

For Cloudlets to attain widespread acceptability, designers and service providers have to propose and present Cloudlet services and frameworks which can enhance the productivity and revenue of a prospective Cloudlet-enabled business, either directly or indirectly. As a first step towards Cloudlet deployment, an organization will analyze the benefits of deploying a Cloudlet in its premises. A business or a client can decide to deploy a Cloudlet in its premises for the following reasons:

As a value added service: A facility or a business, such as a coffee bar or a dine in restaurant, can deploy Cloudlet services to attract customers. They can deploy the Cloudlet and its service, free of cost, at their premises, similar to free Wi-Fi service offered by businesses around the world to give an incentive to the user to visit.

As alternate to Cloud: An organization, already utilizing a public Cloud service, wants to ease load on its Cloud resources by deploying a Cloudlet. This Cloudlet will offer the same service as that of the Cloud and is connected to the Cloud for synchronization.

To provide in-network service: Cloudlet can also be deployed by large telecom providers to provide low cost computing resources to people using mobile devices within a service provider range. This service can be provided at cell-level, without forwarding the traffic to the core-network.

To augment sensor networks: Cloudlets can be used to provide one hop aggregation service to small devices like sensors and smart wearable devices.

Apart from these, many other incentives specific to a particular organization or business exist to deploy a Cloudlet. In any case, before opting to deploy a Cloudlet, an organization will carefully analyze its needs and compare Cloudlet advantages to those of alternative solutions. Cloudlet designers and service providers need to carefully analyze possible objectives for deploying a Cloudlet and devising frameworks and deployment plans, which suites a particular set of prospective clients. Like Cloud

Table 3
Comparison of Cloudlet architectures.

Feature	CACTSE	MOCHA	VM-Based Cloudlet	Cloudlet-Based Multi-Lingual Dictionaries	Cloud-Vision	Cloudlet for IMCA	Pocket Cloudlet	Sensor Cloudlets
Deployment cost	Medium	High	Medium	Low	High	High	Low	High
Implementation complexity	Low	High	High	Low	High	High	Low	High
Management	Centralized – supervised	Centralized – supervised	Centralized – unsupervised	Centralized – unsupervised	Supervised	Supervised – complex	Unsupervised	Distributed – unmanaged
Scalability	High	Medium	Medium	Low	Low	High	High	Medium
Dependence on Internet	High	Medium	Low	Low	High	No	Low	Medium
Support node mobility	No	No	No	No	No	Yes	Yes	Yes
Dynamic configuration of Cloudlet	Partially supported	Not supported	Fully supported	Fully supported	Not supported	Not supported	Fully supported	Fully supported
Network QoS	High	High	Low	High	High		NA	High
Energy saving	High	High	High	High	High	Low	High	Low
Execution cost (as compared to executing on Cloud)	Medium	Low	Low	Low	High	Low	Low	High
Disaster recovery	Yes (limited)	Limited	Limited	No	No	No	Yes	No
Dynamic partitioning	Not required	Fixed + greedy	Not supported	No	Yes fixed + greedy	Fixed	No	No
Offline service availability	Limited	Yes	Yes	Yes	No	Yes	Yes	Yes
Access technology	802.11 Standards	802.11n, Secure Wi-Fi	802.11 Standards	802.11 Standards	802.11n, 802.15.1, high speed radio link	802.11, FlashLinq	NA (Local Cloudlet)	IEEE 802.15.1, 802.11 Standards
Application model	Content delivery network	Big data processing	Application/code offloading	Data processing (dictionary)	Real time processing	Interactive applications	Local cache storage	
Underlying technology	Web services/P2P	GPU MPP architecture	VM overlays	VM overlay	MOCHA/Open CV	NS2	DB driven architecture	Web services, Vendor SDK
Deployment scenario	Social gatherings	Battlefield, face and speech recognition apps	Small enterprises	Information sharing	Face recognition, security	Enterprise applications, video streaming, chat	Web search, social media applications	Urban sensing application, environment monitoring, Google glass

frameworks, an all-in-one solution will be expensive and complex to manage for small businesses seeking to replace the current Cloud-based solutions with a more focused yet simpler solutions. From an organization's point of view, a well defined set of objectives is necessary before assessing different Cloudlet frameworks for possible deployment. Not every Cloudlet framework works well with every possible requirement scenario. So unless objectives are well understood and stated, the choice of an inappropriate Cloudlet framework is bound to produce less than optimal results.

6.2. Feasibility study of Cloudlet frameworks

When an organization is moving towards Cloudlet deployment, it needs to analyze available Cloudlet frameworks and make sure they meet the organization's business requirements. Each Cloudlet framework supports a limited set of services. A comprehensive analysis of business requirements will result in formulating essential Cloudlet services required to fulfill an organization business requirements. Once it is done, the next step is to perform cost estimations, infrastructure requirements study, technology maturity assessment and future support. For an organization, the choice of deploying a particular Cloudlet service and replacing with another Cloud service is always the trickiest part.

Organizations tend to overlook one aspect of it or the other and end up opting for less than the optimal solution. This happens when either business goals or objectives are ambiguous, there is an aggressive marketing by one technology provider, there is incomplete cost analysis, or there exists a dearth of alternate solutions. We have presented a comparison of different Cloudlet frameworks proposed in the literature in Table 3. It is worth mentioning that since this is a relatively new domain very limited performance analysis results are available for deployed systems. It is clear from the analysis that no single Cloudlet is universally fit to provide all kinds of services. The choice of Cloudlet framework requires analysis of user requirements and extensive study of available Cloudlet options.

6.3. Defining standards for Cloudlets

For successful deployment of Cloudlets and its large scale acceptability, standard protocols and data formats are required. All Cloudlet-related tasks, including user and device authentication, data and application exchange, interoperability among different Cloudlet frameworks, seamless migration from one Cloudlet to another, communication between Cloudlet and Cloud services, Cloudlet management and administration, resource discovery, Cloudlet service advertisement, billing and pricing many standards are required. Standards pertaining to Cloudlets are almost non-existent today. The lack of standards related to security, privacy, communication, data storage models, and service discovery is a major barrier towards Cloudlet deployment. However, as the scope and objectives of Cloudlets are limited as compared to a Cloud, the required standards will be relatively simpler. Some possible standards required for Cloudlets are: *Resource Discovery Protocols*: Standards are required for advertising and describing Cloudlet availability, Cloudlet capability, available resources, and services being offered.

Network communication standards: Standards are required for network control signaling, message exchange, cross layer messaging, exchange of user data and meta-data, and exchange of network topology information for device-Cloudlet, inter-Cloudlets, and Cloudlet–Cloud communication.

Authentication standards: Standards for identity verification and authentication are required to establish a trust relationship between client and a Cloudlet, between Cloudlets, and between a Cloudlet and a Cloud.

Application exchange: Standards are required for applications migration among Cloudlets during handover to support device mobility. Similarly, standard application exchange protocols are also required for migrating an application from a Cloudlet to a Cloud.

Billing: Standards for billing and pricing models are required for the successful deployment of the Cloudlets.

Administration: Standards for monitoring, auditing, and reporting are needed.

Programming interface: Standards for developing APIs, etc. for interfacing applications with a Cloudlet framework.

SLAs: Standards are needed for describing Service Level Agreements.

6.4. Pricing and billing models

Developing a billing and pricing model for Cloudlet services will be tricky. The main complication is to define a pricing point which can attract the users to use a Cloudlet instead of their existing Cloud services, and at the same time generate revenue for the organization. Other factors should also be integrated into the pricing model such as ensuring optimal infrastructure consumptions, complementing other services being offered in the same premises/organization and customer's consumption patterns. Cloudlet frameworks can take a note from pricing programs offered by other services offered in the same way. For example businesses like Internet cafes and their customers agree to pay a fixed price per hour for Internet surfing. However, if a user needs extra services such as printing, scanning or telephony, they may pay a higher amount for using the extra services. Unlike Cloud providers, who have started to charge separately for each item; they charge separately for processor, storage, etc., Cloudlet providers may converge their pricing plan to offer an all in one solution. The customers of businesses which are sites for potential Cloudlet deployment (restaurants, clubs, airports, classrooms, shopping malls, etc.) will most likely have common interests at the time they are using the Cloudlet, so differentiating services and charging for each service separately will bear little fruit. On the other hand integrated services and one fixed price can attract customers if the price is justifiable.

There will be other revenue generation models for Cloudlets instead of direct customer charges. Many businesses and organizations deploy services to attract customers to sell their core product. For example, many restaurants offer free Wi-Fi and cable TV, cinema offers free eatables with a movie premier and airlines offer free in-flight entertainment for customers who buy their product. This revenue generation model depends upon two factors. (a) The difference between the amount an organization with a Cloudlet will charge for the product and the amount an organization without the Cloudlet will charge and (b) how attractive the Cloudlet service is to the customers. If an organization charges exorbitantly after installing a Cloudlet, the revenue generation will decrease as it will deter people from purchasing the actual product. On the other hand if the service offered by the Cloudlet is not attractive to potential customers, the introduction of Cloudlets will have little impact on generated revenue. Hence, these two factors need to be considered when deploying Cloudlets and not directly charging for the service. Consideration for devising a pricing model includes but is not limited to the following:

Justifiable pricing: A Cloudlet service provider needs to define a justifiable pricing point for its services. Moreover, a systematic approach is required for the re-evaluation of the proposed scheme.

Integrating return-on-investment model: Traditionally, companies prefer to use an ROI model before taking any investment

decision. Cloudlet pricing models need to be integrated. Any pricing model with hard-to-integrate elements is bound to fail or gain acceptability.

Asset utilization: Any pricing model should encourage efficient resource utilization. It is required that every revenue model should incorporate a resource metric and utilization model for effective monetizing of resources.

Utility style pricing model: Similar to Cloud services, Cloudlet frameworks require a utility style pricing model. Models that can be used for Cloudlet services include, *pay as you go*, *pay for what you use*, *pay less for bulk usage*, and *pay less when you reserve capacity*. These need to be defined and implemented for Cloudlet environments.

Funding future growth: Any defined pricing model should be able to generate enough revenue to fund future growth of the network. No service can last long if it is not able to generate funds for innovation.

Customized pricing model: While utility pricing models will suffice for regular customers, requirements can arise for a customized pricing model based on a customer's unique business model.

6.5. Resource monitoring and utilization measurement

Similar to requirements of Clouds, performance and resource monitoring is an integral part of Cloudlet service maintenance. Moreover these resource monitoring requirements are very different from those needed in legacy systems or for vitalized environment monitoring.

There are numerous solutions available for monitoring Cloud resources however none exist for Cloudlets. Before solutions for resource monitoring in Cloudlets are proposed, researchers need to propose a standard model and decide on a list of parameters of interest that should be covered in an exhaustive performance report which can be produced by a good solution. Performance monitoring in Cloudlets can be broadly categorized into two categories. Monitoring from a provider's view and monitoring from the consumer's view. Cloudlet service providers will be interested in infrastructure performance, whereas customers will be interested in Cloudlet application performance. Requirements for both types of monitoring are discussed as follows:

Infrastructure performance and resource monitoring: A Cloudlet service provider is interested in this kind of report and involves monitoring the performance of resources such as virtual machines, internal network, WAN connectivity, etc. Similar to a Cloud solution, in the Cloudlet environment, individual component performance will not serve to provide the performance measure for the entire Cloudlet. A concept proposed by the Cloud community and equally applicable to the Cloudlet domain is *infrastructure response time* (*IRT*). IRT is the total time taken by an application to place as well as complete a task on a virtual environment. While defining a model for Cloudlet performance monitoring, IRT will be the key metric. Various other metrics include CPU usage, Disk usage, Disk latency, percentage busy, percentage ready, memory swap activity, network bytes in/out, host system state and virtual machine state. The required reports and related metrics are discussed in Table 4.

Application performance: Clients with applications running on the Cloudlet will be interested in measuring the performance of their applications. Unlike infrastructure performance, users are not interested in resources utilized by their applications. Rather, clients are interested in an application's response time. For applications that are designed to run on a Cloudlet, such a measurement mechanism should be built into the application itself.

Table 4
Requirements for infrastructure resource monitoring.

Performance measurements	Metrics
Resource utilization report	Memory utilization, Memory swap activity, Network bytes IN/OUT
Infrastructure response time	Service discovery time, Application response time, Database query response time, Network latency
Virtualization metrics report	Number of VMs in use, Time to initiate a new VM, VM migration overhead
Transaction metrics report	Success percentage of transactions, Transaction count per application
Diagnostics reports	Service down time, Resource bottleneck
End user experience report	Quality of Service measure, Quality of Experience measurement

6.6. Security requirement

For Cloudlet deployment in a LAN environment and acceptability, the biggest challenge will be to define privacy regulations on a shared network medium, which so far, has been considered as a private realm with limited frameworks available for privacy and security considerations. Similar to that in the Cloud, issues related to the wide variation of data privacy regulations from country to country, will also pose some challenges, especially at locations such as airports and transit stations where people with mobile devices from different regions will connect to same Cloudlet.

7. Cloudlet deployment in a LAN: impact and challenges

While a majority of enterprises are rapidly transitioning to the Cloud to deploy their business services, some small and medium enterprises that already possess limited hardware resources struggle with the dilemma of whether or not to move all their services to the Cloud, incurring high cost of leasing Cloud resources as well as purchasing higher Internet bandwidth to avoid WAN latency. For such organizations, Cloudlets are a suitable compromise allowing them to reuse or invest in economical hardware resources that allow them to easily deploy a Cloudlet within their premises. Cloudlets are an affordable solution that can be customized to offer services required by employees or personnel visiting the premises. For example, a coffee shop can deploy a Cloudlet which provides data sharing services for social interaction and cached web pages to its customer, while a Cloudlet deployed in a college IT lab or class room can offer computing resources for code offloading to students working on a project. Similarly bus stations, airports, fast food restaurants, military bases, rails and buses can be equipped with Cloudlets to provide affordable Cloud services to its users. In all these scenarios, the most common mode of accessing services of Cloudlet is over a WLAN.

For any successful Cloudlet deployment the issues faced when deploying the technology in a LAN environment will be the make or break factor. While Cloudlets are simpler to use, more economical than many alternative Cloud services, offer low latency, the first time user may face challenges such as deployment complexity, security and privacy issues and uncertainty due to lack of standardization. It should be noted however that many of these challenges and related solutions have been directly inherited from the Cloud model and will disappear as both technologies become more mature. Some of these challenges are discussed in detail below.

7.1. Technology acquisition and deployment

The decision to adopt a particular Cloudlet solution is immediately followed with the acquisition and deployment of the technology. In case of deploying an open source solution, organizations mostly rely on their own IT resources while with a proprietary framework the appropriate vendor or partner may have to be consulted that may incur an additional cost. Both options in most cases will require investment in extra hardware. In a LAN environment, an organization may have to upgrade or extend its current network using extra servers or network elements such as switches routers and access points to support a Cloudlet. The selection of the Cloudlet framework may dictate the WLAN vendor as well due to vendor specific services offered on different devices. The lack of standardization limits availability of solutions which are agnostic of the underlying LAN technology. As apposed to the Cloud, Cloudlet deployment will almost always require analysis of the deployed LAN and in many cases will require changes in network configuration.

Another important aspect of the deployment phase is that of maintenance and troubleshooting. To avoid downtime a business will require IT staff training or vendor support for resolving network issues with the LAN which are natural consequence of the inevitable increase in the network load due to Cloudlets.

7.2. Integration with Cloud services

A Cloudlet on a LAN may need access to Cloud services over the Internet to download and upload data or to act as a proxy between mobile device on a LAN and Cloud service on the Internet. This integration of a Cloudlet with Cloud services requires a few services to be available on the LAN. Most important is a high speed Internet connection for the Cloudlet to connect to the Cloud server. Secondly, either the Cloudlet should hide itself from the Cloud server and act on the behalf of mobile node (much like a NAT) or the Cloud needs to be aware of the Cloudlet services and is capable of integrating its services with the Cloudlet. It will not be possible to enable all Cloud service providers to recognize various different types of Cloudlet frameworks. A feasible solution is to allow Cloudlets to act like a proxy server. The Cloudlet first tries to serve the mobile device with data already present in the Cloudlet. If it fails to do so, it should forward the request transparently to the Cloud and transfer the response from the Cloud to the mobile node. This implementation is not trivial. A user may not allow a Cloudlet to access protected resources on its behalf such as emails and documents residing on the Cloud service. Secondly many Cloud services utilize protocols for network encryption and certificate exchange to communicate securely with a device. In the presence of a Cloudlet proxy, many of these techniques will not work. Another issue that will be faced by the businesses who are deploying Cloudlets in their LAN is to establish trust among its users. A business with goodwill among its clientage will find it easy but establishing the same trust with Cloud services will be a daunting task. A client can trust his local coffee shop because he knows the owner for a very long time, but a Cloud provider like Amazon EC2 will not trust the Cloudlet deployed at the coffee shop premises and will reject any traffic it conveys on the behalf of its client. Suitable mechanisms or security protocols are needed which allow a user to convey to a third party Cloud service provider to trust a Cloudlet for a particular transaction.

Similarly, Cloudlet frameworks which synchronize some type of data with the Cloud service and cache it locally will need protocols for storing data offline. Data providers such as Google (Maps, Search Results, and Images) use proprietary documents which are only allowed to be viewed online. Storing this data on a system accessible over the LAN will violate many terms and conditions related to data provided by these applications. To overcome this issue, either individual businesses or the Cloudlet developers need to acquire permissions from actual data providers to store their data on the LAN for local use. In both cases a large amount of preliminary research work is needed. Similarly, serving the data on the LAN, which Cloud services provide over the Internet will be another challenging task. Protocols for serving data over Internet are already mature and in common practice. This includes protocols for Service Discovery, Data Exchange, Authentication, Identity Management and Control Messaging. Many of these protocols will need to be modified in order to work in a LAN environment. For example, Key distribution servers deployed on the Internet provide public keys to allow for secure session key exchange for communicating parties which require end-to-end encryption. This mechanism of key distribution within a LAN environment is not workable. Similarly service discovery protocols which rely on directory services will need additional services introduced in the LAN to function properly. To overcome this issue, LAN versions of these protocols need to be introduced by Cloudlet developers requiring modification at different layers of the protocol stack ranging from the network to the application as shown in Table 5.

Table 5
Modifications in the network protocol stack required for Cloudlets.

Service	Internet	Cloudlet (required modification)
Network layer		
Addressing	IPv4 (Public IP), IPv6	IPv4 (Private IP) Support Multicast in LAN
Address resolution	DNS Servers	Local DNS for Cloudlet
Address acquisition	DHCP Server	Local DHCP with static addressing
Routing and path discovery	Routers	NA
Transport layer		
Fragmentation	Routers	Not needed
Congestion and flow control	Intermediate router	LAN version of Internet Protocols required
Proxy, tunneling, traffic filter	Proxy servers, EDGE routers, Firewalls	**Content filtering**: Content filtering and firewalls on the mobile device **Proxy Service**: Require custom protocols
Application layer		
Encryption and certification management	PKI server and certificate authority	LAN version of Internet Protocols required
Data exchange protocols	Online http, ftp, SOAP and DB server	Same as Internet
Service discovery protocols	Directory servers, Service announcement messages, Service query protocols	Service announcement messages Service query protocols

7.3. Security and privacy concerns

Establishing trustworthiness of a Cloudlet will be a major challenge in deployment of Cloudlet frameworks. Cloud frameworks are still struggling to remove the shroud of fog related to security and privacy concerns in storage of personal data in the Cloud. Cloudlets, which will belong to small businesses and organizations who are less interested in investing in security and privacy infrastructure, will find it a more daunting task. In Cloudlets the sharing of a common infrastructure by individual mobile device users will definitely raise eyebrows related to privacy of data. Organizations such as the Cloud Security Alliance (CSA) and European Network and Information Security Agency (ENISA) need to propose and harmonize privacy regulations across different countries. At the same time these agencies need to propose benchmarks to enable users to compare privacy regulations before placing their trust on a deployed Cloudlet.

8. Conclusions

In this paper, we discussed Cloudlet frameworks – a technology becoming increasingly popular because it addresses the issues plaguing Mobile Cloud Computing. We presented a hierarchical taxonomy of Cloudlet models and it was shown how state-of-the-art frameworks map to this taxonomy. Technical details of communication, data staging and computation employed by Cloudlets were presented. Cloudlet architectures were compared at a fine-grained level of data based on their deployment costs, implementation complexity, scalability and underlying technologies used. Noteworthy case studies of Cloudlet frameworks were discussed in detail. As the case for any new technology that is being widely adopted we discussed the limitations of the current standardization efforts, integration with existing services and issues pertaining to security and privacy. As such our survey should be a useful reading from a commercialization, management as well as research perspective. A manager looking to adopt a Cloudlet technology for his business can glean information regarding cost-benefit analysis and deployment scenarios. A researcher can benefit from the challenges and current open issues that this work discusses. The drive to have computation as a service via use of powerful data centers and high-speed networks coupled with the consumer demand for mobility using hand-held devices, the need for reliable, cost-effective Cloudlet technology will be increasing in the coming decade. We hope this study will be a first step in getting a grip on some of the issues pertaining to this game-changing paradigm.

References

Abadi DJ. Data management in the cloud: limitations and opportunities. IEEE Data Eng Bull 2009;32(1):3–12.

Abolfazli S, Sanaei Z, Ahmed E, Gani A, Buyya R. Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges. CoRR, vol. abs/1306.4956, 2013.

Achanta V, Sureshbabu N, Thomas V, Sahitya M, Rao S. Cloudlet-based multi-lingual dictionaries. In: 2012 3rd international conference on services in emerging markets (ICSEM), December 2012. p. 30–6.

Ahmed E, Akhunzada A, Whaiduzzaman M, Gani A, Ab Hamid SH, Buyya R. Network-centric performance analysis of runtime application migration in mobile cloud computing. Simul Model Pract Theory 2015a;50:42–56.

Ahmed E, Gani A, Khan MK, Buyya R, Khan SU. Seamless application execution in mobile cloud computing: motivation, taxonomy, and open challenges. J Netw Comput Appl 2015b;52:154–72.

Ahmed E, Gani A, Sookhak M, Ab Hamid SH, Xia F. Application optimization in mobile cloud computing: motivation, taxonomies, and open challenges. J Netw Comput Appl 2015c;52:52–68.

Ahuja SP, Rolli AC. Exploring the convergence of mobile computing with cloud computing. Netw Commun Technol 2012;1(1):97.

Aljabre A. Cloud computing for increased business value. Int J Bus Soc Sci 2012;3(1):6.

Angin P, Bhargava Bharat K. Real-time mobile-cloud computing for context-aware blind navigation, 2012.

Barbosa FP, Charão AS. Impact of pay-as-you-go cloud platforms on software pricing and development: a review and case study. In: Computational science and its applications—ICCSA. Salvador, Bahia, Brazil. 2012. Springer, 2012. p. 404–17.

Bassi A, Geir Horn. Internet of things in 2020. European Commission and the RFID working group of EPoSS. Technical Report [Online]. Available: ⟨http://www.smart-systems-integration.org/public/documents/publications/Internet-of-Things_in_2020_EC-EPoSS_Workshop_Report_2008_v3.pdf⟩.

Bari MF, Boutaba R, Esteves R, Granville LZ, Podlesny M, Rabbani MG, et al. Data center network virtualization: a survey. IEEE Commun Surv Tutor 2013;15(2):909–28.

Benson T, Akella A, Shaikh A, Sahu S. Cloudnaas: a cloud networking platform for enterprise applications. In: Proceedings of the 2nd ACM symposium on cloud computing, SOCC '11. New York, NY, USA: ACM; 2011. p. 8:1–13. [Online]. Available: ⟨http://doi.acm.org/10.1145/2038916.2038924⟩.

Bozakov Z. Architecture and algorithms for virtual routers as a service. In: 2011 IEEE 19th international workshop on quality of service (IWQoS). San Jose, CA, USA. IEEE, 2011. p. 1–3.

Cardellini V, De Nito Personé V, Di Valerio V, Facchinei F, Grassi V, Lo Presti F, et al. A game-theoretic approach to computation offloading in mobile cloud computing. Technical report, 2013. Available online at ⟨http://www.optimizationonline.org/DBHTML/2013/08/3981.html⟩.

Chandramouli B, Claessens J, Nath S, Santos I, Zhou W. Race: real-time applications over cloud-edge. In: Proceedings of the 2012 ACM SIGMOD international conference on management of data, SIGMOD '12. New York, NY, USA: ACM; 2012. p. 625–8. [Online]. Available: ⟨http://doi.acm.org/10.1145/2213836.2213916⟩.

Chebrolu K, Raman B, Sen S. Long-distance 802.11b links: performance measurements and experience. In: Proceedings of the 12th annual international conference on mobile computing and networking, MobiCom '06. New York, NY, USA: ACM; 2006. p. 74–85. [Online]. Available: ⟨http://doi.acm.org/10.1145/1161089.1161099⟩.

Chen Y, Liu B, Chen Y, Li A, Yang X, Bi J. Packetcloud: an open platform for elastic in-network services. In: Proceedings of the 8th ACM international workshop on mobility in the evolving Internet architecture, MobiArch '13. New York, NY, USA: ACM; 2013. p. 17–22. [Online]. Available: ⟨http://doi.acm.org/10.1145/2505906.2505910⟩.

Chu C-K, Zhu W-T, Han J, Liu J, Xu J, Zhou J. Security concerns in popular cloud storage services. IEEE Pervasive Comput 2013;12(October (4)):50–7.

Chun B-G, Ihm S, Maniatis P, Naik M, Patti A. Clonecloud: elastic execution between mobile device and cloud. In: Proceedings of the 6th conference on computer systems, EuroSys '11. New York, NY, USA: ACM; 2011. p. 301–14. [Online]. Available: ⟨http://doi.acm.org/10.1145/19664451966473⟩.

Clemons J, Zhu H, Savarese S, Austin T. Mevbench: a mobile computer vision benchmarking suite. In: 2011 IEEE international symposium on workload characterization (IISWC), November 2011. p. 91–102.

Cloud computing—a primer ⟨http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_12-3/123_cloud1.html⟩. accessed: 2014-04-18.

Cloudmagic app ⟨https://www.cloudmagic.com/⟩, accessed: 2014-09-18.

Cloud storage and unlimited online backup ⟨http://www.livedrive.com/⟩. accessed: 2014-04-18.

Costa P, Migliavacca M, Pietzuch P, Wolf AL. Naas: network-as-a-service in the cloud. In: Proceedings of the 2nd USENIX conference on hot topics in management of Internet, cloud, and enterprise networks and services, Hot-ICE, 2012. [Online]. Available: ⟨https://www.usenix.org/system/files/conference/hot-ice12/hotice12-final29.pdf⟩.

Cuervo E, Balasubramanian A, Cho D-k, Wolman A, Saroiu S, Chandra R, et al. Maui: making smartphones last longer with code offload. In: Proceedings of the 8th international conference on mobile systems, applications, and services. San Fransisco, CA, USA. ACM; 2010. p. 49–62.

Deka GC. A survey of cloud database systems. IT Professional 2014(2):50–7.

Devanathan Nandhagopal NM, Ravichandran S, Malpani S. Vmware and xen hypervisor performance comparisons in thick and thin provisioned environments. 4 May 2012. [Online]. Available: ⟨http://morse.colorado.edu/~tlen5710/12s/VMware.pdf⟩.

Dey S, Liu Y, Wang S, Lu Y. Addressing response time of cloud-based mobile applications. In: Proceedings of the first international workshop on mobile cloud computing & networking, MobileCloud '13. New York, NY, USA: ACM; 2013. p. 3–10. [Online]. Available: ⟨http://doi.acm.org/10.1145/2492348.2492359⟩.

Dolphin browser ⟨https://www.dolphin.com/⟩. accessed: 2014-09-18.

Dropbox, cloud storage and backup ⟨https://www.dropbox.com/business/cloud-storage-and-backup⟩. accessed: 2014-04-18.

Echeverria S, Root J, Bradshaw B, Lewis G. On-demand vm provisioning for cloudlet-based cyber-foraging in resource-constrained environments. In: 2014 6th international conference on mobile computing, applications and services (MobiCASE), November 2014. p. 116–24.

Falaki H, Mahajan R, Kandula S, Lymberopoulos D, Govindan R, Estrin D. Diversity in smartphone usage. In: Proceedings of the 8th international conference on mobile systems, applications, and services, MobiSys '10. New York, NY, USA: ACM; 2010. p. 179–94. [Online]. Available: ⟨http://doi.acm.org/10.1145/1814433.1814453⟩.

Fesehaye D, Gao Y, Nahrstedt K, Wang G. Impact of cloudlets on interactive mobile cloud applications. In: 2012 IEEE 16th international enterprise distributed object computing conference (EDOC), September 2012. p. 123–32.
Flickr photo sharing. ⟨https://www.flickr.com/⟩. accessed: 2014-04-18.
Foster I, Zhao Y, Raicu I, Lu S. Cloud computing and grid computing 360-degree compared. In: Grid computing environments workshop, 2008. GCE '08, November 2008. p. 1–10.
Foursquare – location-based social networking website for mobile devices ⟨http://foursquare.com⟩, accessed: 2014-04-18.
Gao J, Sivaraman A, Agarwal N, Li H, Peh L-S. Diploma: consistent and coherent shared memory over mobile phones. In: 2012 IEEE 30th international conference on computer design (ICCD). Montreal, Canada. IEEE, 2012. p. 371–8.
Gigaspaces xap. [Online]. Available: ⟨http://www.gigaspaces.com/xap-in-memory-caching-scaling/datagrid⟩.
Google cloud storage, cloud backup, cloud database ⟨https://cloud.google.com/products/cloud-storage/⟩, accessed: 2014-04-18.
Gordon AW, Lu P. Low-latency caching for cloud-based web applications. In: Proceedings of the 6th international workshop on networking meets databases (NetDB11), Athens, Greece, Citeseer, 2011.
Grolinger K, Capretz M, Mezghani E, Exposito E. Knowledge as a service framework for disaster data management. In: 2013 IEEE 22nd international workshop on enabling technologies: infrastructure for collaborative enterprises (WETICE), June 2013. p. 313–8.
Ha K, Pillai P, Lewis G, Simanta S, Clinch S, Davies N, et al. The impact of mobile multimedia applications on data center consolidation. In: 2013 IEEE international conference on cloud engineering (IC2E), March 2013. p. 166–76.
Hassan MA, Bhattarai K, Wei Q, Chen S. Pomac: properly offloading mobile applications to clouds. Energy 2014;25:50.
Huerta-Canepa G, Lee D. A virtual cloud computing provider for mobile devices. In: Proceedings of the 1st ACM workshop on mobile cloud computing & services: social networks and beyond, MCS '10. New York, NY, USA: ACM; 2010. p. 6:1–5. [Online]. Available: ⟨http://doi.acm.org/10.1145/1810931.1810937⟩.
Jararweh Y, Tawalbeh L, Ababneh F, Dosari F. Resource efficient mobile computing using cloudlet infrastructure. In: 2013 IEEE 9th international conference on mobile ad-hoc and sensor networks (MSN). Dalian, China. IEEE, 2013. p. 373–7.
Ji C, Li Y, Qiu W, Awada U, Li K. Big data processing in cloud computing environments. In: 2012 12th international symposium on pervasive systems, algorithms and networks (ISPAN). San Marcos, Texas, USA. IEEE, 2012. p. 17–23.
Khalaj A, Lutfiyya H, Perry M. A proxy-based mobile computing infrastructure. In: 2012 3rd FTRA international conference on mobile, ubiquitous, and intelligent computing (MUSIC). Vancouver, Canada. IEEE, 2012. p. 21–8.
Koukoumidis E, Lymberopoulos D, Strauss K, Liu J, Burger D. Pocket cloudlets. In: 2011 16th international conference on Architectural support for programming languages and operating systems. Newport Beach, California, USA. ACM, 2011. pp.171-184 [Online]. Available: ⟨http://dblp.uni-trier.de/db/conf/asplos/asplos2011.html#KoukoumidisLSLB11⟩.
Kumar K, Lu Y-H. Cloud computing for mobile users. Computer 2011;PP(99):51-56.
Kumar K, Liu J, Lu Y-H, Bhargava B. A survey of computation offloading for mobile systems. Mob Netw Appl 2013;18(February (1)):129–40 [Online]. Available: ⟨http://dx.doi.org/10.1007/s11036-012-0368-0⟩.
Lago P. Cloudlet-based cyber-foraging in resource-constrained environments. September 2013. [Online]. Available: ⟨http://www.slideshare.net/patricia_lago/g-lewis-cloudlet-based-cyberforagingvupptx⟩.
Lewis GA, Echeverría S, Simanta S, Bradshaw B, Root J. Cloudlet-based cyber-foraging for mobile systems in resource-constrained edge environments. In: Companion proceedings of the 36th international conference on software engineering, ICSE Companion 2014. New York, NY, USA: ACM; 2014. p. 412–5. [Online]. Available: ⟨http://doi.acm.org/10.1145/2591062.2591119⟩.
Lewis G, Echeverría S, Simanta S, Bradshaw B, Root J. Tactical cloudlets: moving cloud computing to the edge. In: 2014 IEEE military communications conference (MILCOM). Baltimore, MD, USA. IEEE, 2014. p. 1440–6.
Liu W-L. Cloud storage performance and security analysis with hadoop and gridftp. 2013.
Liu F, Shu P, Jin H, Ding L, Yu J, Niu D, et al. Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications. IEEE Wirel Commun 2013;20(June (3)):14–22.
Liu J, Ahmed E, Shiraz M, Gani A, Buyya R, Qureshi A. Application partitioning algorithms in mobile cloud computing: taxonomy, review and future directions. J Netw Comput Appl 2015;48:99–117.
Malathy G, Somasundaram R, Duraiswamy K. Performance improvement in cloud computing using resource clustering. J Comput Sci 2013;9(6):671.
Manzalini A, Saracco R, Buyukkoc C, Chemouil P, Kukliński S, Gladisch A, et al. Software-defined networks for future networks and services: main technical challenges and business implications. IEEE SDN4FNS Whitepaper based on the IEEE Workshop SDN4FNS. 2014. [Online]. (Available: http://sdn.ieee.org/articles-publications.html).
Matias J, Jacob E, Sanchez D, Demchenko Y. An openflow based network virtualization framework for the cloud. In: Proceedings of the 2011 IEEE third international conference on cloud computing technology and science, CLOUDCOM '11. Washington, DC, USA: IEEE Computer Society; 2011. p. 672–8. [Online]. Available: ⟨http://dx.doi.org/10.1109/CloudCom.2011.104⟩.
McCarthy D, Malone P, Hange J, Doyle K, Robson E, Conway D, et al. Personal cloudlets: implementing a user-centric datastore with privacy aware access control for cloud-based data platforms. In: Proceedings of the first international workshop on technical and legal aspects of data privacy. Florence, Italy. IEEE Press, 2015. p. 38–43.
Nelson AJ, Dinolt GW, Michael JB, Shing M-T. A security and usability perspective of cloud file systems. In: 2011 6th international conference on system of systems engineering, 2011.
Openi framework, objectives ⟨http://www.openi-ict.eu/objectives/⟩, accessed: 2014-04-18.
Opera browser ⟨https://www.opera.com/⟩, accessed: 2014-09-18.
Othman M, Hailes S. Power conservation strategy for mobile computers using load sharing. ACM Mob Comput Commun Rev 1998;2:44–50.
Ou S, Wu Y, Yang K, Zhou B. Performance analysis of fault-tolerant offloading systems for pervasive services in mobile wireless environments. In: IEEE international conference on communications, 2008. ICC'08. Beijing, China. IEEE; 2008. p. 1856–60.
Plattner H. A common database approach for oltp and olap using an in-memory column database. In: Proceedings of the 2009 ACM SIGMOD international conference on management of data, SIGMOD '09. New York, NY, USA: ACM; 2009. p. 1–2. [Online]. Available: ⟨http://doi.acm.org/10.1145/1559845.1559846⟩.
Qing W, Zheng H, Ming W, Haifeng L. Cactse: Cloudlet aided cooperative terminals service environment for mobile proximity content delivery. Commun China 2013;10(June (6)):47–59.
Rajesh S, Swapna S, Reddy PS. Data as a service (daas) in cloud computing. Global J Comput Sci Technol 2012;12(11-B):55.
Rudenko A, Reiher P, Popek GJ, Kuenning GH. Saving portable computer battery power through remote process execution. SIGMOBILE Mob Comput Commun Rev 1998;2(January (1)):19–26 [Online]. Available: ⟨http://doi.acm.org/10.1145/584007.584008⟩.
Sarwar S, Ul-Qayyum Z, Malik OA. A hybrid intelligent system to improve predictive accuracy for cache prefetching. Expert Syst Appl 2012;39(2):1626–36.
Satyanarayanan M. Pervasive computing: vision and challenges. IEEE Personal Commun 2001;8(August (4)):10–7.
Satyanarayanan M. Mobile computing: the next decade. In: Proceedings of the 1st ACM workshop on mobile cloud computing & services: social networks and beyond, MCS '10. New York, NY, USA: ACM; 2010. p. 5:1–6. [Online]. Available: ⟨http://doi.acm.org/10.1145/1810931.1810936⟩.
Satyanarayanan M. The role of cloudlets in hostile environments. In: Proceeding of the 4th ACM workshop on mobile cloud computing and services, MCS '13. New York, NY, USA: ACM; 2013. p. 1–2. [Online]. Available: ⟨http://doi.acm.org/10.1145/2482981.2483793⟩.
Satyanarayanan M, Bahl P, Caceres R, Davies N. The case for vm-based cloudlets in mobile computing. IEEE Pervasive Comput 2009;8(October (4)):14–23.
Satyanarayanan M, Lewis G, Morris E, Simanta S, Boleng J, Ha K. The role of cloudlets in hostile environments. IEEE Pervasive Comput 2013;12(4):40–9.
Satyanarayanan M, Simoens P, Xiao Y, Pillai P, Chen Z, Ha K, et al. Edge analytics in the Internet of things. IEEE Pervasive Comput 2015(2):24–31.
Shao B, Wang H, Li Y. Trinity: a distributed graph engine on a memory cloud. In: Proceedings of the 2013 ACM SIGMOD international conference on management of New York, USA. 2013. p. 505–16.
Sherry J, Hasan S, Scott C, Krishnamurthy A, Ratnasamy S, Sekar V. Making middleboxes someone else's problem: network processing as a cloud service. ACM SIGCOMM Comput Commun Rev 2012;42(4):13–24.
Shivarudrappa D, Chen M, Bharadwaj S. Cofa: automatic and dynamic code offload for android. University of Colorado, Boulder, 2011.
Soyata T, Muraleedharan R, Funai C, Kwon M, Heinzelman W. Cloud-vision: real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. In: 2012 IEEE symposium on computers and communications (ISCC), July 2012. p. 000059–000066.
Soyata T, Muraleedharan R, Langdon J, Funai C, Ames S, Kwon M, et al. Combat: mobile-cloud-based compute/communications infrastructure for battlefield applications, 2012. p. 84030K–84030K-13. [Online]. Available: http://dx.doi.org/10.1117/12.919146.
Terzo O, Ruiu P, Bucci E, Xhafa F. Data as a service (daas) for sharing and processing of large data collections in the cloud. In: 2013 7th international conference on complex, intelligent, and software intensive systems (CISIS), July 2013. p. 475–80.
The nist definition of cloudcomputing ⟨http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf⟩, accessed: 2014-04-18.
Understanding cloud storage models. [Online]. Available: ⟨http://www.infoworld.com/article/2871290/cloud-computing/understanding-cloud-storage-models.html⟩.
Vaughan-Nichols S. Openflow: the next generation of the network?. Computer 2011;44(August (8)):13–5.
Verbelen T, Simoens P, De Turck F, Dhoedt B. Cloudlets: bringing the cloud to the mobile user. In: Proceedings of the 3rd ACM workshop on mobile cloud computing and services, MCS '12. New York, NY, USA: ACM; 2012. p. 29–36. [Online]. Available: ⟨http://doi.acm.org/10.1145/2307849.2307858⟩.
Verbelen T, Simoens P, De Turck F, Dhoedt B. Leveraging cloudlets for immersive collaborative applications. IEEE Pervasive Comput 2013;12(4):30–8.
Vimeo video sharing ⟨https://www.vimeo.com/⟩, accessed: 2014-04-18.
Wang S, Dey S. Cloud mobile gaming: modeling and measuring user experience in mobile wireless networks. SIGMOBILE Mob Comput Commun Rev 2012;16(July (1)):10–21 [Online]. Available: ⟨http://doi.acm.org/10.1145/2331675.2331679⟩.
Wang K, Rao J, Xu C-Z. Rethink the virtual machine template. In: Proceedings of the 7th ACM SIGPLAN/SIGOPS international conference on virtual execution environments, VEE '11. New York, NY, USA: ACM; 2011. p. 39–50. [Online]. Available: ⟨http://doi.acm.org/10.1145/1952682.1952690⟩.

Wei Y, Blake MB. Service-oriented computing and cloud computing: challenges and opportunities. IEEE Internet Comput 2010;14(6):72–5.

Wu X, Tavildar S, Shakkottai S, Richardson T, Li J, Laroia R, et al. Flashlinq: a synchronous distributed scheduler for peer-to-peer ad hoc networks. IEEE/ACM Trans Netw 2013;21(August (4)):1215–28.

Xu S, Zhang W. Knowledge as a service and knowledge breaching. In: Proceedings of the 2005 IEEE international conference on services computing, SCC '05, vol. 01. Washington, DC, USA: IEEE Computer Society; 2005. p. 87–94. [Online]. Available: http://dx.doi.org/10.1109/SCC.2005.60.

Yang Z, Niyato D, Wang P. Offloading in mobile cloudlet systems with intermittent connectivity. IEEE Trans Mob Comput 2015;PP(99): 2516-2529.

Yu R, Zhang Y, Gjessing S, Xia W, Yang K. Toward cloud-based vehicular networks with efficient resource management. IEEE Netw 2013;27(5):48–55.

Zhang Y, Niyato D, Wang P, Tham C-K. Dynamic offloading algorithm in intermittently connected mobile cloudlet systems. In: 2014 IEEE international conference on communications (ICC), June 2014. p. 4190–5.

Zhao B, Xu Z, Chi C, Zhu S, Cao G. Mirroring smartphones for good: a feasibility study. In: Mobile and ubiquitous systems: computing, networking, and services, vol. 73. Berlin, Heidelberg: Springer; 2012. p. 26–38. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-29154-8_3.

Zhou Z, Huang D. Efficient and secure data storage operations for mobile cloud computing. In: Network and service management (cnsm), 2012 8th international conference and 2012 workshop on systems virtualization management (svm), October 2012. p. 37–45.

10 of the most useful cloud databases. [Online]. Available: ⟨http://www.networkworld.com/article/2162274/cloud-computing/10-of⟩.