International Journal of Cloud Computing

ISSN online: 2043-9997 - ISSN print: 2043-9989
https://www.inderscience.com/ijcc

A discovery and selection protocol for decentralised cloudlet systems

Dilay Parmar, Padmaja Joshi, Udai Pratap Rao, A. Sathish Kumar, Ashwin Nivangune

DOI: 10.1504/IJCC.2023.10054985

Article History:
Received: 23 April 2019
Accepted: 07 April 2020
Published online: 27 March 2023

A discovery and selection protocol for decentralised cloudlet systems

Dilay Parmar*

Computer Engineering Department,
S.V. National Institute of Technology,
Surat, Gujarat, 395-007, India
Email: dilayparmar@gmail.com
*Corresponding author

Padmaja Joshi

Centre for Development of Advanced Computing,
Mumbai, Maharashtra, 400-049, India
Email: padmaja@cdac.in

Udai Pratap Rao

Computer Engineering Department,
S.V. National Institute of Technology,
Surat, Gujarat, 395-007, India
Email: upr@coed.svnit.ac.in

A. Sathish Kumar

Centre for Development of Advanced Computing,
Mumbai, Maharashtra, 400-049, India
Email: asathish@cdac.in

Ashwin Nivangune

Unisys India Private Limited,
Bangalore, Karnataka, 560-035, India
Email: ashwin4234@gmail.com

Abstract: Cloudlets help in overcoming latency issue of clouds in mobile cloud computing to offload the computing tasks. Communication protocols are important part of the implementation of cloudlet-based systems for the mobile-cloudlet-cloud environment. In this work, an approach for communication between entities in a decentralised cloudlet-based system is proposed. For that purpose, a cloudlet discovery protocol which is used for discovering cloudlets in Wi-Fi vicinity of mobile devices is proposed. A selection algorithm for selecting the suitable cloudlet from available discovered cloudlets is also proposed. Our proposed selection algorithm uses infrastructure specific criterion for selection decision, which makes the algorithm more generic to use.

Keywords: cloudlet; mobile cloud computing; MCC; discovery; selection.

Reference to this paper should be made as follows: Parmar, D., Joshi, P., Rao, U.P., Kumar, A.S. and Nivangune, A. (2023) 'A discovery and selection protocol for decentralised cloudlet systems', *Int. J. Cloud Computing*, Vol. 12, No. 1, pp.1–22.

Biographical notes: Dilay Parmar is currently a Research Scholar and pursuing his PhD in Computer Engineering from the S.V. National Institute of Technology, Surat. Prior to this, he has worked as an Assistant Professor at the P.P. Savani University, Visiting Faculty at S.V. National Institute of Technology, Surat, and Research Intern at C-DAC, Mumbai. His research interests include areas related to location privacy and mobile cloud computing. He has authored a number of papers in international conferences from renowned publishers. He received his MTech in Computer Engineering from the S.V. National Institute of Technology, Surat, India.

Padmaja Joshi is currently working with the C-DAC, Mumbai as the Senior Director and heads Software Engineering Division. She currently leads research and projects in the area of mobile computing, e-governance, blockchain, authentication and software engineering. Her work is published in many renowned international journals and conferences. She pioneered workshop on reverse engineering (IWRE) and a track on 'mobile software engineering' in an international conference ISEC. She also had been a keynote speaker at various IEEE and ACM forums. She holds a PhD in the area of Software Engineering from the IIT Bombay, India.

Udai Pratap Rao is currently associated with the Department of Computer Engineering in S.V. National Institute of Technology, Surat, Gujarat, since August 2007. He obtained his PhD in Computer Engineering from the S.V. National Institute of Technology, Surat. His research interests include information security and privacy, location-based privacy, data mining, big data analytics and distributed computing. He delivered a number of keynote and plenary lectures at several national and international conferences. He has published extensively in reputed journals and refereed conference proceedings.

A. Sathish Kumar is currently working with the C-DAC, Mumbai as a Senior Technical Officer. He is currently working in the area of mobile application development and e-governance. He is doing research in the area of mobile computing.

Ashwin Nivangune is currently working as the Lead Systems Engineer with the Unisys India Private Limited. He has 12+ years of experience in the area of system programming and carrying out research in domains like microservice architecture, distributed computing, mobile cloud computing, critical infrastructure protection, network security and software-defined networking. His work in these domains is published in many renowned international journals and conferences. He has completed his MTech in Information Technology from the NMIMS (Deemed to be) University, India.

This paper is a revised and expanded version of a paper entitled 'Discovery and selection mechanism of cloudlets in a decentralized MCC environment', presented at Third IEEE/ACM International Conference on Mobile Software Engineering and Systems, Austin, TX, USA, 16–17 May 2016.

1 Introduction

Today, smartphones are used to perform multiple activities such as working on mails, documents, images, anytime anywhere. However, still mobile devices (MDs) are not able to handle compute-intensive tasks like speech recognition, natural language processing, augmented reality and decision-making (Satyanarayanan et al., 2009). MDs leverage cloud computing for their compute intensive tasks.

It is observed that mobile hardware is always resource poor compared to static client and server hardware. The major concern for MDs is dealing with weight, size, battery life, and heat dissipation which results in less powerful computational resources such as processor speed, memory and disk capacity (Satyanarayanan, 1996).

Resource poverty in MDs is a major limitation, due to which applications requiring high computation resources and battery power cannot run on mobile hardware. Applications like face recognition, language translation and applications involving human cognition are very useful with MDs. However, these applications require a high level of processing and energy, which is currently not available with MDs.

Cloud computing may help in overcoming the computation limitation posed by resource-poor MDs (Joshi et al., 2015). A MD can execute the resource intensive task on the distant cloud and get back the results from the cloud. It solves the resource poverty issue but has some other issues associated with it. These issues are discussed in the next section.

The rest of the paper is organised as follows. Section 2 discusses about issues pertaining to mobile cloud computing (MCC). Section 3 discusses the work done in this domain and contribution through this paper. Section 4 covers architecture and classification of cloudlet systems. In this section, various scenarios for communication between MD and cloudlet are also covered. Our proposed protocol is given in Section 5. Section 6 gives details regarding experimentation. Results are presented in Section 7. Section 8 concludes the work. Finally, Section 9 provides the future directions.

2 MCC challenges

For MCC environment, latency is a major drawback. Latency is based on the distance between the MD and the cloud, available network bandwidth, processing time taken by the MD, processing time taken by cloud infrastructure, and time spent on the network, i.e., network latency (Messinger and Lewis, 2013).

Applications demanding crisp response time suffer a lot due to the latency issue. If the cloud resources are very far from the MDs, then wide area network (WAN) delay will also be increased resulting into poor interactive response. Tasks such as web browsing might be working fine with latency but, tasks such as augmented reality become slow (Satyanarayanan et al., 2009). Satyanarayanan et al. (2009) state that WAN latency may not improve because the modern research for networking mainly focuses on security and manageability issues. As a solution to the large WAN delay, data centres (DCs) must be brought closer, i.e., cloud service providers (CSPs) should build lots of DCs around the world with appropriate strategic placement. Building DCs is expensive and capital intensive. Instead of making of DCs around the world, some other alternative smarter approach must be taken.

MDs can leverage cloud systems for offloading its resource intensive tasks. However, cloud computing involves high WAN delay. Thus, for latency sensitive applications, leveraging cloud is not a good solution. Cloudlets are devices with better resources that are available in the nearby vicinity of MDs and are ready to take certain load of mobile application computation. Cloudlets reduce the large WAN delay caused by the distant cloud. With this, mobile users can meet the need of real-time interactive response by low latency, one-hop, high bandwidth wireless access to the cloudlet (Satyanarayanan et al., 2009).

The communication protocol between a MD that is interested in offloading its work and the cloudlet which is ready to take the task plays a very vital role. Due to the considerable differences between cloudlet and cloud environment, communication protocols used for cloud environment may not be suitable for cloudlet environment.

A novel communication protocol is proposed and implemented for a decentralised cloudlet environment in this paper. In this paper, firstly various cloudlet system architectures are elaborated. A cloudlet discovery and selection protocol is then proposed for the MD and the cloudlet communication.

3 Related work

Cyber foraging is a technique to make resource-poor MDs leverage surrogate machines for their computation-intensive tasks (Satyanarayanan, 2001). Surrogate is expected to have more powerful hardware resources. They execute the offloaded task on behalf of MDs and return back the computed results to MDs. MCC is an amalgamation of mobile computing and cloud computing (Fernando et al., 2013; Khan et al., 2014). In Ahmed et al. (2015), the process of offloading an application to the cloud is presented. Issues pertaining to application offloading is also shown in Ahmed et al. (2015). In Palaniappan and Ramasamy (2017), a survey on mobile offloading techniques is presented. In Pati et al. (2018), conflict-free scheduling of tasks for MCC is proposed. Applicability of the cloud in reduction of power usage of MDs is discussed in Abolfazli et al. (2014) and Thanapal and Durai (2018). Most of this work is based on remote cloud offloading. However, for latency sensitive applications, cloudlets provide good performance as compared to distant cloud (Satyanarayanan et al., 2009).

Most of the approaches put emphasis on the usage of cloudlets. Soyata et al. (2012) presented a cloudlet-based solution for real-time face recognition with MDs. Lin et al. (2013) used cloudlet in a graphics application. Fesehaye et al. (2012) presented a cloudlet-based solution for interactive mobile applications like collaborative chat and file editing. To the best of our knowledge, most of the approaches do not provide details about, how the discovery and selection of cloudlets are done. These approaches assume that the MD has discovered a cloudlet and communication with cloudlet is already established.

According to Satyanarayanan (2001), the discovery of surrogates (e.g., cloudlets) is a major challenge for cyber foraging. Selection of a cloudlet is another challenge. Some approaches deal with multi cloudlet scenario, where multiple cloudlets are candidates for offloading. In Jia et al. (2017), optimal placement and allotment of cloudlet for the metropolitan area is presented. In Mukherjee et al. (2016), an approach for cloudlet selection is proposed which uses latency and power as selection criterions. Tawalbeh et al. (2015) discuss about multiple cloudlets' deployment for achieving a reduction in

power consumption. In Liu and Fan (2018), a cloudlet selection model based on mixed integer linear programming is presented. Roy et al. (2017) presented an application-aware selection method for multi-cloudlets environment.

In our work (Parmar et al., 2016), we have considered multiple cloudlets' discovery and selection. Extended abstract for that was presented in Parmar et al. (2016). In this work, the entire protocol (Parmar et al., 2016) is discussed at length along with experimentation details and results. The detailed explanation of cases of protocol along with required preliminaries is also presented.

4 Background

4.1 Terminologies

The terminology followed in this paper is as follows:

1 *MD:* MDs are a limited resource and limited energy devices that want to transfer their compute/resource intensive tasks to cloud. Here, MDs can be smartphones or other wearable cognitive assistance devices like Google glasses.

2 *CSP:* It surrogates services for MDs. CSPs have enough resources to handle MDs' resource requests.

3 *Internet service provider (ISP):* ISP provides internet to MDs for requesting the CSP.

4 *Base transceiver station (BTS) and Wi-Fi access point:* MDs connect to ISP with the help of either BTS or Wi-Fi access point. Initially, MDs may connect to BTS or Wi-Fi access point to get connected to internet for transferring their resource requests to cloud.

5 *Cloudlet devices (CLDs):* Cloudlet is a new architectural element used by nearby MDs. It is connected to cloud. When MDs want to offload their tasks, they can offload it to nearby cloudlets. If no cloudlets are present in the vicinity of MDs, then MDs can send their tasks to the distant cloud or otherwise in the worst case, it will perform the tasks on itself. CLDs can be workstations, server machines, tablet computers, etc. that are available or deployed explicitly in the nearby vicinity. In general, CLDs are resource-rich devices compared to the MDs.

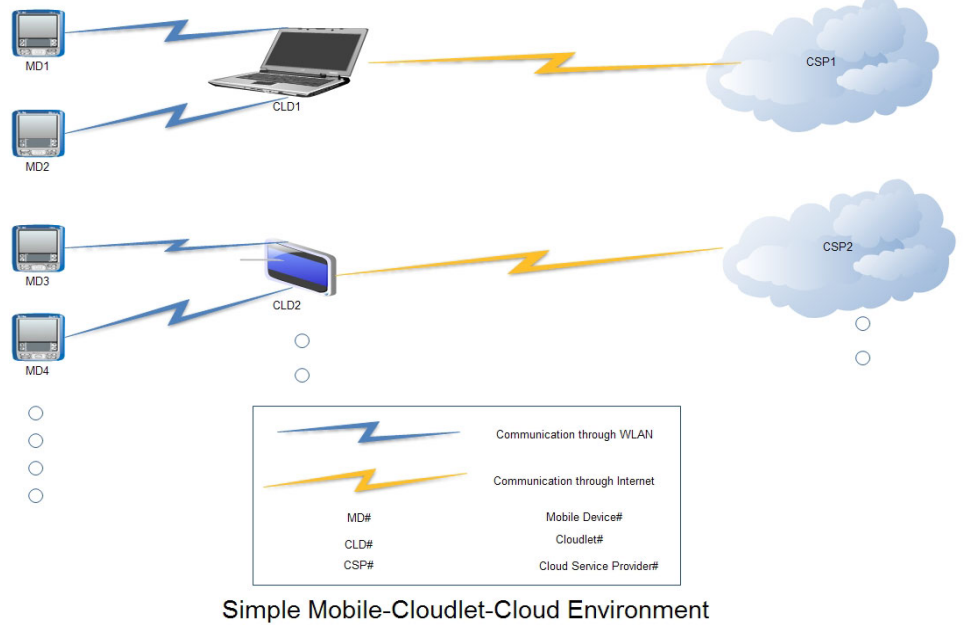
6 *Centralised registry server (CRS):* CRS keeps a record of available cloudlets in a particular area. MDs may use CRS for discovering CLDs. Whenever a MD wants to offload its task to cloudlets, it contacts this registry server to get information about available cloudlets. New cloudlets can first register their services into the CRS so that MDs can identify appropriate cloudlet with the help of the registry server.

   CRS should perform heart-beat check-up operations on cloudlets registered with it to know the availability of cloudlets. Cloudlets have to keep their information up-to-date in the registry server. CRS is useful in the environment where multiple cloudlets are deployed within the nearby area and information of all the cloudlets can be made available to the CRS.

Figure 1 depicts the basic architecture of MCC environment. It is evident that the MDs are directly communicating with the CSP for work offloading. Figure 2 depicts the basic architecture of mobile cloudlet cloud environment.

Figure 1 Basic MCC architecture (see online version for colours)

Figure 2 Basic mobile-cloudlet-cloud architecture (see online version for colours)



In the basic MCC environment, WAN latency limitation is inevitable where MDs transfer their tasks to the surrogate cloud with the help of the internet. As a solution to that limitation, Satyanarayanan et al. (2009) introduced cloudlets.

4.2 Classification of cloudlet-based systems

Lewis and Lago (2015) have classified surrogate discovery as a centralised registry-based (local or cloud) and broadcast-based approach. For cloudlet-based systems, it can be classified into two types based on the approach used for cloudlet identification: centralised systems and decentralised systems.

4.2.1 Centralised system

In this type of system, MDs discover cloudlets with the help of the CRS. Cloudlets of a particular area are connected to a centralised server. This centralised server acts as a registry/directory of cloudlets. All the available cloudlets first register themselves into the registry, and then MDs requesting cloudlet resources consult registry server for getting information of available cloudlets.

Rawadi et al. (2014) and Artail et al. (2015) discuss about providing cloudlet services to MDs with inter-cloudlet communication in their research paper. It is an example of centralised cloudlet-based system. In this, all cloudlets are connected to the root server with the help of the internet. Here, cloudlets can communicate with other cloudlets with the help of LAN.

Centralised systems are also called as directory-based systems because of the importance of directory/registry in the system.

- Pros of centralised cloudlet-based systems
  1. Management of cloudlet resources scattered over the region is done centrally.
  2. Easy to implement compared to decentralised cloudlet-based system.
  3. Control of authority, i.e., CRS can be easily managed and service description is also consistent.
  4. Handoff management can be done by the centralised server, when MDs are moving from one cloudlet area to other cloudlet area.
  5. Offloading decision can be taken by the centralised server as it is having all the information of cloudlets with it. This takes burden from resource poor MDs.
- Cons of centralised cloudlet-based systems
  1. A centralised system is suitable for university type of environment where, a centralised server can manage information of departmental cloudlets. It is not suitable if number of cloudlets is more and steadily increasing.
  2. Central point of failure-directory/registry server, i.e., if the CRS is vanished, then no MDs can get information of available cloudlets and can get connected to it.

4.2.2 Decentralised systems

In this type of system, MDs discover cloudlets in the vicinity of them without making use of any directory or registry mechanisms. Here, MDs use multicasting mechanisms to find available cloudlets in their vicinity.

Messinger and Lewis (2013) use multicast-based cloudlet discovery mechanism in their implementation, which is an example of a decentralised cloudlet-based system. The discovery mechanism is provided by the JmDNS library, which is a pure Java implementation of multicast DNS and zeroconf framework. Cloudlet server registers a service and publishes its information. The MDs use JmDNS for discovering the services that are published in the zero configuration multicast group. Those discovered services will be shown to MDs.

- Pros of decentralised cloudlet-based system
  1 No single point of failure because there is no CRS.
  2 For discovering cloudlets MDs do not rely on any kind of registry mechanism.
  3 Direct peer to peer communication with cloudlet.
  4 No restriction on scalability of the system.
- Cons of decentralised cloudlet-based system
  1 Network floods out with large number of multicast request packets.
  2 Service description is not consistent everywhere since, all cloudlets use their own service description format.
  3 Offloading decision must be taken by MD itself since no central authority involved who can take decision on behalf of MD.
  4 Handoff management must be done by MDs itself; since, all cloudlets are isolated with each other due to no common node involved in it like registry node.

4.2.3 Remarks over centralised vs. decentralised cloudlet-based system

1 Centralised cloudlet-based system is suitable for the environment where few cloudlets are deployed in the area. The centralised server can have information about all the cloudlets available in that area, e.g., cloudlets placed in a college or a university where multiple departments have cloudlets deployed in their departmental premises and a central registry server placed on the campus. MDs can directly contact the central server for cloudlet information and select appropriate cloudlet for them. Selection of cloudlet and handoff management tasks can be handled by central registry server.

2 Decentralised cloudlet-based system is suitable for a distributed environment where cloudlets work independently and do not rely on any centralised entity. MDs can directly identify cloudlets in their region with P2P communication and they can negotiate with cloudlets for using them as surrogates, e.g., cloudlet placed in a coffee shop or in a mall. MDs directly identify cloudlets in its area without using any kind of directory mechanism. Offloading decision is taken by MDs from the list of available cloudlets with it. Handoff management may be done by MDs itself since cloudlets are isolated with each other.

4.3 Challenges in cloudlet environment

- *Discovery of cloudlets:* In decentralised cloudlet-based systems, identification of cloudlets is not so easy. MDs have to find available cloudlets in their proximity by itself only. For that purpose, MDs have to use discovery protocols based on multicasting. Selection of appropriate discovery protocol with appropriate customisation is required.
- *Handoff management in cloudlet:* After offloading tasks to cloudlets if the MD moves from one cloudlet's coverage area to other cloudlet's coverage area, at that time, proper handoff mechanisms must be introduced to handle the mobility issue.

  Offloading decision:

  1 Where to offload?

    If multiple cloudlets are available in MD's vicinity, then out of available cloudlets which cloudlet should be selected for offloading tasks based on what parameters?

  2 Who will take offloading decision?

    The decision of whether to offload the task to cloudlet or to cloud must be taken by some entity in the cloudlet-based environment. For centralised and decentralised systems, this decision may be taken by different entities.
- *Trust and security issues:* MDs must confidently transfer their tasks to cloudlet. For that, cloudlet must be secured and trustworthy device. Proper security measures must be adapted for having a secure cloudlet-based experience.

Out of the identified challenges we have worked on discovery and selection part, i.e., where to offload part of offloading decision for decentralised cloudlet systems.

4.4 Communication model under consideration

In this paper, decentralised cloudlet infrastructure is assumed, where MDs, cloudlets, and CSP are considered as part of the ecosystem. CLDs are battery powered MDs which provide surrogate services. Here, the following types of communication take place:

- *Communication between MD and cloudlet:* Initially, cloudlets broadcast its services in its region. MDs scan the region for available cloudlets in its vicinity and get the information of available cloudlets. At this place, our discovery and selection protocol is required to be used.
- *Communication between cloudlet and cloudlet:* In an ideal scenario, the MD transfers the task to cloudlet and waits for the result. Cloudlet executes the task, and results are sent to the MD. Here, mobility cases of MD and cloudlet are required to be considered. If MDs are moving from one cloudlet area to another cloudlet area after transferring tasks, at that time, an inter-cloudlet communication protocol is required by means of which results can be sent to the MD.
- *Communication between cloudlet and cloud:* If the cloudlet is not having the cloud portion/code with it, then cloudlet will communicate with cloud and get the cloud

portion/code from it. After executing the downloaded code from the cloud, the MD can then offload its task to cloudlet.

- *Communication between MD and cloud:* If cloudlets are not available in the MD's vicinity, then the MD will have to communicate with cloud in order to offload their task.

Our proposed protocol is used in the communication of MD and CLD. Our protocol assumes that a common list of applications' name and ID is available with MD and cloudlets. The MD receives a list of available application IDs from cloudlets along with other details. Subsequently, a selection of appropriate cloudlet is done based on our selection algorithm.

5 Proposed protocol for cloudlet discovery and selection

The proposed discovery protocol identifies available cloudlets in the MD's Wi-Fi vicinity without being connected to any of the cloudlet's local network. An algorithm for selecting suitable cloudlet for offloading tasks is also proposed. The proposed selection algorithm uses infrastructure specific requirements for task offloading by the MD, which makes the overall selection mechanism more generic.

5.1 Cloudlet discovery

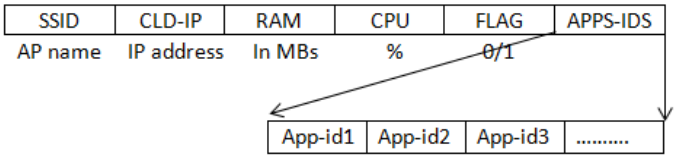
In the discovery process, identification of cloudlets in Wi-Fi range is done by the MD through Wi-Fi P2P service. With multicast advertisement, cloudlet publishes its information. For discovering cloudlets, the scan operation is performed by the MD by using a multicast query. Android API android.net.wifi.p2p.nsd is employed in this protocol, which provides support for Bonjour and UPnP service discovery (UPnP Discovery, n.d.) over Wi-Fi medium. The main advantage of Wi-Fi P2P service discovery is, it grants MDs in discovering the cloudlets without being connected to any of the cloudlet's local network (Using Wi-Fi P2P for Service Discovery, n.d.). Cloudlet publishes itself as a service so that the MD can identify it with the service discovery framework. Here, service information contains the actual information of the cloudlet which includes:

- SSID of cloudlet's Wi-Fi access point
- cloudlet's IP address
- cloudlet's unoccupied/free RAM
- cloudlets's free CPU percentage
- flag for identifying CLD as battery driven or power supplied device
- application IDs of apps present on the cloudlet.

A frame in the form of a character array is made from these parameters of cloudlet information as shown in Figure 3. For publishing the required information, the Bonjour service discovery protocol is referred. Bonjour protocol (Bonjour Overview, n.d.) can transfer 100–200 bytes of information in the form of a text record, which is sufficient for

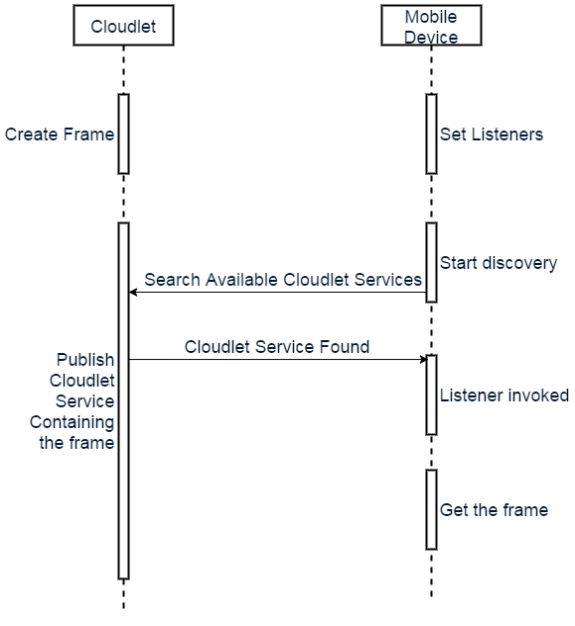
transferring the cloudlet information. The frame is passed in the text record format. To save the amount of data exchange over a network, we have assigned single byte integer values to applications. This mapping is known to the mobile and the cloudlet. Cloudlets are not elastic in nature like a cloud. Hence, a number of applications running/installed on a cloudlet at any given instance are limited. For current exploration, we are limiting the number of application published over Wi-Fi to ten. Note that these are published by the cloudlet; however, the cloudlet may have more than ten applications installed in it.

Figure 3 Frame structure of cloudlet publishing



Source: Parmar et al. (2016)

Figure 4 Sequence diagram showing interactions of a cloudlet and a MD



Source: Parmar et al. (2016)

Cloudlet discovery protocol

- At cloudlet end:
  1. Make a string map and add frame having information of cloudlet.
  2. Make an object *WiFiP2PDnsSdServiceInfo* and provide instance name of service, type of service, and the map created in Step 1 to this object.

3 Put the cloudlet service for discovery. Create an object of *Wi-FiP2pManager* and call method *addLocalService* with the object created in Step 2 as one of the parameters. This enables the cloudlet service to be discovered over Wi-Fi.

- At mobile end:
  1 Set the listeners on for handling service discovery response from cloudlet. It will be called upon discovery of any cloudlet service.

    Set *setDnsSdResponseListeners* which contains two more listeners as parameters:

    a *DnsSdServiceResponseListener* – used in retrieving service information

    b *DnsSdTxtRecordListener* – used in retrieving the frame.
  2 Make service discovery request which will enable discovery process for different cloudlet services. Call method *addServiceRequest*.
  3 Start cloudlet service discovery. Call method *discoverServices*.

  Interactions of MD and cloudlet are depicted in the form of a sequence diagram in Figure 4.

5.2 Cloudlet selection

After the discovery phase, MD has gained information of the available cloudlets in terms of service information. Now, the selection of cloudlet for offloading is done, from the list of available cloudlets. It is done based on its requirement of the application offloading.

Initially, the application requirements (RAM, CPU) are compared with cloudlet specifications, which were obtained as per the cloudlet structure.

Direct elimination for selection of cloudlet can be applied for the cases where cloudlets are not having enough memory and CPU. Since, the performance of the application may not be well, if such cloudlets are selected.

For achieving better performance, cloudlets with the already deployed application should be considered. However, in absence of application on cloudlet, downloading of application from cloud should be done. It may not serve the purpose of offloading, since the execution of an application on the MD may take less time compared to the whole process of downloading and then using. For this reason, the list of application IDs is made available in the discovery part. Hence, the next criterion of cloudlet selection is the availability of desired application on the cloudlet.

Another major criterion considered for selection is type of power supply used by cloudlets. In the selection process, attention is given to the device which has its own power supply in contrast to the battery powered device, e.g., server machine or workstation running as the cloudlet will be the first choice as compared to the tablet.

Selection of cloudlet is done by the MD through comparison of application's parameters with cloudlets' parameters.

- Application parameters:
  1 *R-RAM:* maximum RAM requirement of application in MB
  2 *R-CPU:* maximum CPU requirement of application in percentage
  3 *APP-ID:* application ID.

- Cloudlet parameters:
  1 *RAM[i]:* free RAM in i^{th} cloudlet
  2 *CPU[i]:* free CPU percentage in i^{th} cloudlet
  3 *Powered[i]:* flag indicating that if the i^{th} cloudlet is battery powered or has power supply
    a *Powered[i] = 0* shows that the cloudlet is battery powered
    b *Powered[i] = 1* shows that the cloudlet has power supply
  4 *APPsIDs[i]:* applications IDs of applications which are already deployed on the i^{th} cloudlet.

Selection of a cloudlet is made based on the following priority:

1 Priority-1

$$\text{R-RAM} \leq \text{RAM[i]} \;\&\; \text{R-CPU} \leq \text{CPU[i]} \;\&\; \text{APP-ID} \in \text{APPsIDs[i]} \;\&\; \text{Powered[i]} = 1$$

The cloudlet having the app already installed and running on the power supply will be the first choice for selection.

2 Priority-2

$$\text{R-RAM} \leq \text{RAM[i]} \;\&\; \text{R-CPU} \leq \text{CPU[i]} \;\&\; \text{APP-ID} \in \text{APPsIDs[i]} \;\&\; \text{Powered[i]} = 0$$

The cloudlet having the app already installed but, running on the battery will be the second choice for selection.

3 Priority-3

$$\text{R-RAM} \leq \text{RAM[i]} \;\&\; \text{R-CPU} \leq \text{CPU[i]} \;\&\; \text{APP-ID} \notin \text{APPsIDs[i]} \;\&\; \text{Powered[i]} = 1$$

The cloudlet not having the app already installed and running on the power supply will be the third choice for selection. Cloudlet has to download the application first and then provide service to the MD.

4 Priority-4

$$\text{R-RAM} \leq \text{RAM[i]} \;\&\; \text{R-CPU} \leq \text{CPU[i]} \;\&\; \text{APP-ID} \notin \text{APPsIDs[i]} \;\&\; \text{Powered[i]} = 0$$

The cloudlet not having the app already installed and running on the battery will be the last choice for selection.

Cloudlet selection algorithm in terms of a flowchart is depicted in Figure 5 and Figure 6. The MD has the list of discovered cloudlets. Cloudlet selection algorithm is applied on this list. Variable *Select* will hold the cloudlet-ID of the selected cloudlet. Initially, variable *Select* holds a null value. Variable *i* is used for indexing the cloudlet entries in the list, last indicates the index of the last cloudlet entry in the discovered cloudlet list.

Firstly required RAM and CPU resources of the application are compared with the resource information of cloudlets from the list. If the cloudlet has the required RAM and CPU resources, then further processing are applied, else the next cloudlet entry is checked.

Figure 5 Cloudlet selection algorithm part A

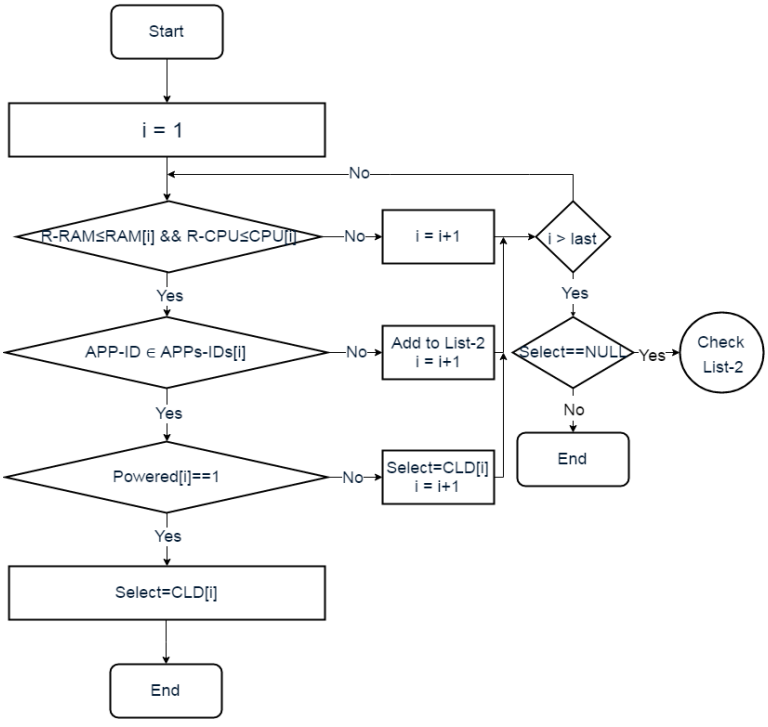


Figure 6 Cloudlet selection algorithm part B

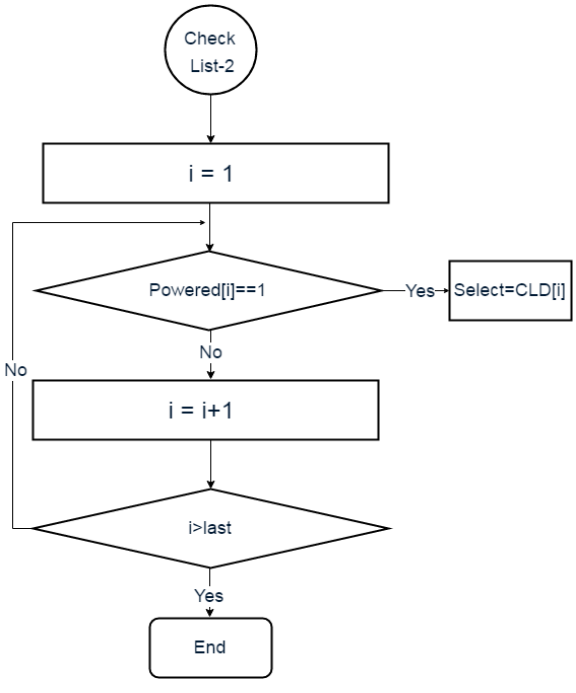


Table 1 Selection algorithm cases and expected results

No.	*CLD1*			*CLD2*			*Select*	*Note*
	R-RAM ≤ RAM[1] && R-CPU ≤ CPU[1]	*APP_ID ∈ APPS_IDs[1]*	*POWERED*	*R-RAM ≤ RAM[2] && R-CPU ≤ CPU[2]*	*APP_ID ∈ APPS_IDs[2]*	*POWERED*		
1	N	-	-	N	-	-	Null	CPU and RAM requirement: not satisfied for CLD1 and CLD2 Application installed: do not care Power supply: do not care Selection: none
2	N	-	-	Y	-	-	CLD2	CPU and RAM requirement: CLD1 not satisfied and CLD2 satisfied Application installed: do not care Power supply: do not care Selection: CLD2
3	Y	-	-	N	-	-	CLD1	CPU and RAM requirement: CLD1 satisfied and CLD2 not satisfied Application installed: do not care Power supply: do not care Selection: CLD1
4	Y	Y	-	Y	N	-	CLD1	CPU and RAM requirement: CLD1 satisfied and CLD2 satisfied Application installed: CLD1 installed and CLD2 not installed Power supply: do not care Selection: CLD1
5	Y	N	-	Y	Y	-	CLD2	CPU and RAM requirement: CLD1 satisfied and CLD2 satisfied Application installed: CLD1 not installed and CLD2 installed Power supply: do not care Selection: CLD2
6	Y	Y	Y	Y	Y	N	CLD1	CPU and RAM requirement: CLD1 satisfied and CLD2 satisfied Application installed: CLD1 installed and CLD2 installed Power supply: CLD1 hard wired and CLD2 battery powered Selection: CLD1

Table 1 Selection algorithm cases and expected results (continued)

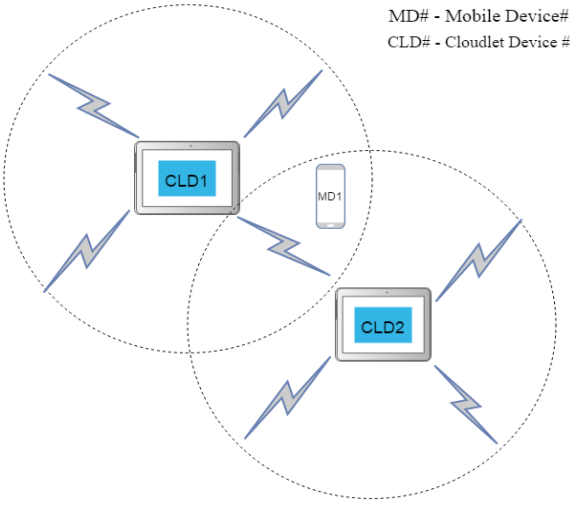
No.	*CLD1*			*CLD2*			*Select*	*Note*
	R-RAM ≤ *RAM[1]* && *R-CPU* ≤ *CPU[1]*	*APP_ID* ∈ *APPS_IDs[1]*	*POWERED*	*R-RAM* ≤ *RAM[2]* && *R-CPU* ≤ *CPU[2]*	*APP_ID* ∈ *APPS_IDs[2]*	*POWERED*		
7	Y	Y	N	Y	Y	Y	CLD2	CPU and RAM requirement: CLD1 satisfied and CLD2 satisfied Application installed: CLD1 installed and CLD2 installed Power supply: CLD1 battery powered and CLD2 hard wired Selection: CLD2
8	Y	Y	N	Y	Y	N	CLD2	CPU and RAM requirement: CLD1 satisfied and CLD2 satisfied Application installed: CLD1 installed and CLD2 installed Power supply: CLD1 battery powered and CLD2 battery powered Selection: CLD2 (as per algorithm)
9	Y	N	Y	Y	N	N	CLD1	CPU and RAM requirement: CLD1 satisfied and CLD2 satisfied Application installed: CLD1 not installed and CLD2 not installed Power supply: CLD1 hard wired and CLD2 battery powered Selection: CLD1
10	Y	N	N	Y	N	Y	CLD2	CPU and RAM requirement: CLD1 satisfied and CLD2 satisfied Application installed: CLD1 not installed and CLD2 not installed Power supply: CLD1 battery powered and CLD2 hard wired Selection: CLD2
11	Y	N	N	Y	N	N	CLD1	CPU and RAM requirement: CLD1 satisfied and CLD2 satisfied Application installed: CLD1 not installed and CLD2 not installed Power supply: CLD1 battery powered and CLD2 battery powered Selection: CLD1 (as per algorithm)

When a cloudlet matching the Priority-1 criteria is identified, then the selection process stops and declares that cloudlet as the suitable cloudlet for offloading. No further checking is done for other cloudlet entries. If the cloudlet matching the Priority-2 criteria is identified, then that cloudlet is considered as the selected cloudlet until no Priority-1 fulfilling cloudlet is identified from the list.

A new list, i.e., list-2 is maintained for the cloudlet entries of the original list who do not have the app already installed on it. At the end of the cloudlet selection process as shown in Figure 5, variable *Select* gives the cloudlet-ID of the selected cloudlet. If *Select* is null, then the cloudlet entries stored in list-2 are considered for the cloudlet selection. Out of the cloudlet entries stored in list-2, the cloudlet having the power supply is selected over the battery-powered cloudlets, which is shown in Figure 6. In case if no cloudlets are having the application installed on it and running on battery power, then the first cloudlet from list-2 is selected since the maximum requirement in terms of CPU and RAM resources of application is satisfied by all the cloudlets who are part of list two.

Table 1 depicts the stated evaluation cases and expected results for selection among two cloudlets. In Table 1, 'N' represents that particular condition holds false, 'Y' represents that particular condition holds true, and '-' represents do not care condition, i.e., it can hold true or false, no effect on results. R-RAM and R-CPU represents the required RAM and CPU to offload the application. RAM[i] and CPU[i] represents available RAM and CPU at cloudlet 'i'. POWERED = N represents cloudlet is battery powered and POWERED = Y represents cloudlet is a wired device. APP_ID represents the application-id of the application to be offloaded. APPS_IDs[i] represents already installed applications on cloudlet 'i'.

Figure 7 Experimental setup (see online version for colours)



6 Experimentation

As shown in Figure 7, three devices were used in the experiment. Two devices were used as CLDs and one device was used as a MD seeking discovery and selection of cloudlet

for offloading tasks. All three devices were Android OS-based devices. The specifications of devices used for experimentation are given in Table 2. Two MDs running on Android L, and Android M was used as CLDs. One device was used as MD seeking discovery and selection of cloudlet. Both MD 1 and MD 3 ran cloudlet publish application. MD 2 ran the discovery application. All three devices were Wi-Fi P2P enabled devices.

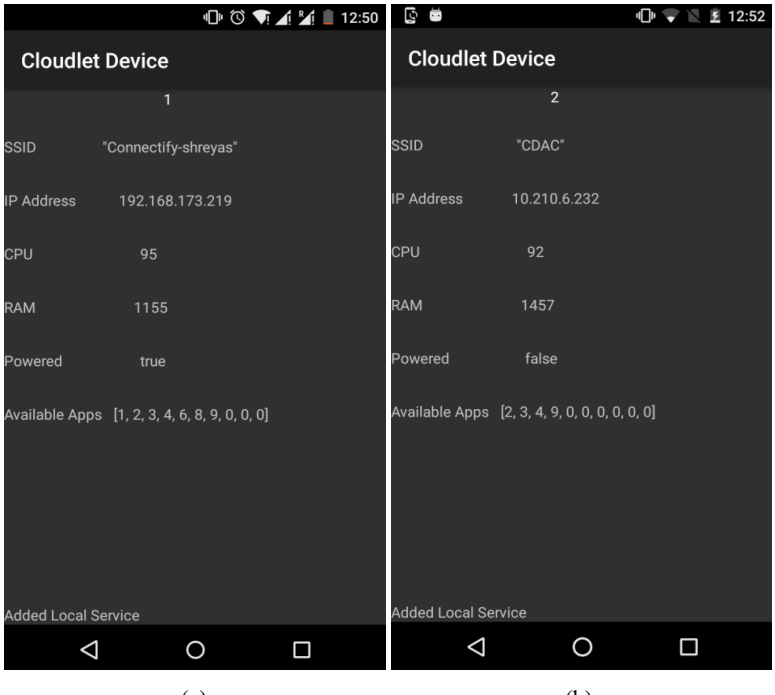
Table 2 Device specifications

Device name	*RAM*	*OS*
Mobile device 1	2 GB	Android 6.0 (Marshmallow)
Mobile device 2	1 GB	Android 5.1 (Lollipop)
Mobile device 3	2 GB	Android 5.1.1 (Lollipop)

7 Results and analysis

Figure 8 shows two cloudlets running the cloudlet application and publishing its information. Figure 9 shows discovered cloudlets with its information on MD. Table 3 shows the results obtained for different cases of input. As shown in Figure 8 and Figure 9, the information obtained by the MD is same as the information published by the CLDs. The expected results and obtained results are same for both the proposals, i.e., for discovery protocol and for selection algorithm.

Figure 8 Cloudlets publishing its information, (a) cloudlet-1 publishing its information (b) cloudlet-2 publishing its information (see online version for colours)

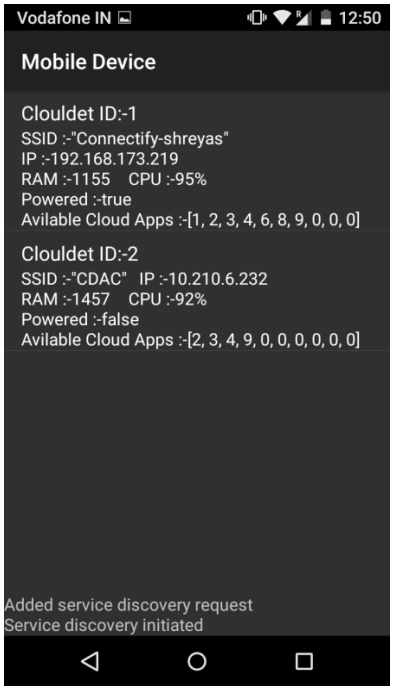


(a) (b)

Table 3 Test results for selection among two cloudlets

Test case no.		*1*	*2*	*3*	*4*	*5*	*6*	*7*
App parameters	R-RAM	1,600	1,400	800	900	850	750	700
	R-CPU	50%	50%	50%	55%	40%	60%	40%
	APP_ID	4	3	6	2	9	5	7
CLD1	Free RAM	1,145	1,173	1,174	1,183	1,059	1,096	1,086
	Free CPU	93%	86%	93%	96%	98%	98%	89%
	APPS_IDs	1, 2, 3, 4, 6, 8, 9	1, 2, 3, 4, 6, 8, 9	1, 2, 3, 4, 6, 8, 9	1, 2, 3, 4, 6, 8, 9	1, 2, 3, 4, 6, 8, 9	1, 2, 3, 4, 6, 8, 9	1, 2, 3, 4, 6, 8, 9
	POWERED	True	True	True	True	False	True	False
CLD2	Free RAM	1,444	1,447	1,440	1,444	1,422	1,424	1,408
	Free CPU	95%	97%	97%	88%	63%	92%	96%
	APPS_IDs	2, 3, 4, 9	2, 3, 4, 9	2, 3, 4, 9	2, 3, 4, 9	2, 3, 4, 9	2, 3, 4, 9	2, 3, 4, 9
	POWERED	False	False	False	False	False	False	False
Expected result (selection)		Null	CLD2	CLD1	CLD1	CLD2	CLD1	CLD1
Actual result (selection)		Null	CLD2	CLD1	CLD1	CLD2	CLD1	CLD1

Figure 9 MD discovering two cloudlets with its information



8 Conclusions

In this paper, a cloudlet discovery and selection protocol is presented for the decentralised cloudlet environment. In order to present the proposed protocol, a broader view on centralised cloudlet systems and decentralised cloudlet systems is also presented. The proposed selection algorithm considers different aspects of selecting a cloudlet for offloading tasks. At the same time, the algorithm is more generic to use since it uses infrastructure specific parameters. So, for different kinds of applications giving different services, the same algorithm can be used. The applicability of the protocol is also demonstrated using case studies.

9 Future directions

The work in this paper was a first step towards establishing protocols for code offloading in decentralised cloudlet environment. It can further be extended:

1 Currently, the proposed discovery protocol and selection algorithm are implemented for android platform-based devices. This can be extended for other platforms also such as Apple iOS, Blackberry, Windows Phone OS, etc. to reach the mass.

2 In case, if a MD moves from one cloudlet region to another cloudlet region after offloading the tasks then results obtained by previously connected cloudlet are

required to be sent to the MD. For this purpose, a cloudlet to cloudlet communication protocol is required. For a more robust cloudlet experience, this handoff protocol can be designed and implemented in the future.

References

Abolfazli, S., Sanaei, Z., Ahmed, E., Gani, A. and Buyya, R. (2014) 'Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges', *IEEE Communications Surveys & Tutorials*, Vol. 16, No. 1, pp.337–368.

Ahmed, E., Gani, A., Khan, M.K., Buyya, R. and Khan, S.U. (2015) 'Seamless application execution in mobile cloud computing: motivation, taxonomy, and open challenges', *Journal of Network and Computer Applications*, Vol. 52, pp.154–172.

Artail, A., Frenn, K., Safa, H. and Artail, H. (2015) 'A framework of mobile cloudlet centers based on the use of mobile devices as cloudlets', in *Proceedings of the Twenty-ninth IEEE International Conference on Advanced Information Networking and Applications*, IEEE, pp.777–784.

Bonjour Overview (n.d.) [online] https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/NetServices/Introduction.html\#//appleref/doc/uid/TP40002445-SW1 (accessed 9 January 2019).

Fernando, N., Loke, S.W. and Rahayu, W. (2013) 'Mobile cloud computing: a survey', *Future Generation Computer Systems*, Vol. 29, No. 1, pp.84–106.

Fesehaye, D., Gao, Y., Nahrstedt, K. and Wang, G. (2012) 'Impact of cloudlets on interactive mobile cloud applications', in *Proceedings of the Sixteen IEEE International Enterprise Distributed Object Computing Conference*, IEEE, pp.123–132.

Jia, M., Cao, J. and Liang, W. (2017) 'Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks', *IEEE Transactions on Cloud Computing*, Vol. 5, No. 4, pp.725–737.

Joshi, P., Nivangune, A., Kumar, R., Kumar, S., Ramesh, R., Pani, S. and Chesum, A. (2015) 'Understanding the challenges in mobile computation offloading to cloud through experimentation', in *Proceedings of the Second ACM International Conference on Mobile Software Engineering and Systems*, IEEE, pp.158–159.

Khan, A.U.R., Othman, M., Madani, S.A. and Khan, S.U. (2014) 'A survey of mobile cloud computing application models', *IEEE Communications Surveys & Tutorials*, Vol. 16, No. 1, pp.393–413.

Lewis, G. and Lago, P. (2015) 'Architectural tactics for cyber-foraging: results of a systematic literature review', *Journal of Systems and Software*, Vol. 107, pp.158–186.

Lin, T., Zhou, K. and Wang, S. (2013) 'Cloudlet-screen computing: a client-server architecture with top graphics performance', *International Journal of Ad Hoc and Ubiquitous Computing*, Vol. 13, No. 2, pp.96–108.

Liu, L. and Fan, Q. (2018) 'Resource allocation optimization based on mixed integer linear programming in the multi-cloudlet environment', *IEEE Access*, Vol. 6, pp.24533–24542.

Messinger, D. and Lewis, G. (2013) *Application Virtualization as a Strategy for Cyber Foraging in Resource-constrained Environments*, Technical Report CMU/SEI-2013-TN-007, Software Engineering Institute, Carnegie Mellon University.

Mukherjee, A., De, D. and Roy, D.G. (2016) 'A power and latency aware cloudlet selection strategy for multi-cloudlet environment', *IEEE Transactions on Cloud Computing*, Vol. 7, No. 1, pp.141–154.

Palaniappan, N. and Ramasamy, S. (2017) 'A survey on procedures dealing with mobile offloading schemes', *International Journal of Mobile Network Design and Innovation*, Vol. 7, Nos. 3–4, pp.178–188.

Parmar, D., Kumar, A.S., Nivangune, A., Joshi, P. and Rao, U.P. (2016) 'Discovery and selection mechanism of cloudlets in a decentralized MCC environment', in *Proceedings of the Third IEEE/ACM International Conference on Mobile Software Engineering and Systems*, ACM, pp.15–16.

Pati, B., Panigrahi, C.R. and Sarkar, J.L. (2018) 'CETM: a conflict-free energy efficient transmission policy in mobile cloud computing', *International Journal of Communication Networks and Distributed Systems*, Vol. 20, No. 2, pp.129–142.

Rawadi, J., Artail, H. and Safa, H. (2014) 'Providing local cloud services to mobile devices with inter-cloudlet communication', in *Proceedings of the Seventeenth IEEE Mediterranean Electrotechnical Conference*, IEEE, pp.134–138.

Roy, D.G., De, D., Mukherjee, A. and Buyya, R. (2017) 'Application-aware cloudlet selection for computation offloading in multi-cloudlet environment', *The Journal of Supercomputing*, Vol. 73, No. 4, pp.1672–1690.

Satyanarayanan, M. (1996) 'Fundamental challenges in mobile computing', in *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing*, ACM, pp.1–7.

Satyanarayanan, M. (2001) 'Pervasive computing: vision and challenges', *IEEE Personal Communications*, Vol. 8, No. 4, pp.10–17.

Satyanarayanan, M., Bahl, P., Caceres, R. and Davies, N. (2009) 'The case for vm-based cloudlets in mobile computing', *IEEE Pervasive Computing*, Vol. 8, No. 4, pp.14–23.

Soyata, T., Muraleedharan, R., Funai, C., Kwon, M. and Heinzelman, W. (2012) 'Cloud-vision: real-time face recognition using a mobile-cloudlet-cloud acceleration architecture', in *2012 IEEE Symposium on Computers and Communications (ISCC)*, IEEE, pp.000059–000066.

Tawalbeh, L., Jararweh, Y., Dosari, F. et al. (2015) 'Large scale cloudlets deployment for efficient mobile cloud computing', *Journal of Networks*, Vol. 10, No. 1, pp.70–77.

Thanapal, P. and Durai, M.S. (2018) 'Energy saving offloading scheme for mobile cloud computing using cloudsim', *International Journal of Advanced Intelligence Paradigms*, Vol. 10, Nos. 1–2, pp.45–62.

UPnP Discovery (n.d.) [online] https://docs.microsoft.com/en-us/previous-versions/windows/embedded/ms885488(v=msdn.10) (accessed 11 January 2019).

Using Wi-Fi P2P for Service Discovery (n.d.) [online] http://developer.android.com/training/connect-devices-wirelessly/nsd-wifi-direct.html (accessed 11 January 2019).