International Journal of Computer (IJC)
ISSN 2307-4523 (Print & Online)
http://ijcjournal.org/

A Dynamic Application Partitioning and Offloading Framework to Enhance the Capabilities of Transient Clouds Using Mobile Agents

Tiako Fani Ndambomve[a*], Felicitas Mokom[b], Kolyang Dina Taiwe[c]

[a,b,c]*LaRI Lab, University of Maroua, P.O. Box 814 Maroua, Cameroon*
[a,b]*School of Information Technology, Catholic University Institute of Buea, P.O. Box 563 Buea, Cameroon*
[b]*IEEE Computational Intelligence Society Member*
[b]*Association for Computing Machinery (ACM) Member*
[a]*Email: fanimichele12@gmail.com, tiakofani@cuib-cameroon.net.*
[b]*Email: fmokom.dev@gmail.com, fmokom@cuib-cameroon.net*
[c]*Email: dtaiwe@yahoo.fr*

Abstract

Mobile cloud computing has emerged as a prominent area of research, a natural extension of cloud computing that proposes to offer solutions for enhancing the capabilities of smart mobile devices commonly plagued by resource constraints. As one of its promising models, transient clouds aim to address the internet connectivity shortfall inherent in most solutions through the formation of ad hoc networks by devices in close proximity, then the offloading some computations (Cyber Foraging) to the created cloud. However, transient clouds, at their current state, have several limitations, concerning their expansion on a local network having a large number of devices and the management of the instability of the network due to the constant mobility of the devices. Another issue is the fact code partitioning and offloading are not addressed to fit the need of such networks, thereby rendering the distributed computing mechanism barely efficient for the Transient Cloud. In this study, we propose a transient cloud-based framework that exploits the use of multi-agent systems, enabling a dynamic partitioning and offloading of code, and facilitating the movement and the execution of code partition packets in a multi-hop ad-hoc mesh network. When created and deployed, these intelligent mobile agents operate independently or collaboratively and adapt to the continual entry and exit of devices in the neighbourhood. The integration of these trending concepts in distributed computing within a framework offers a new architecture for resource-sharing among cooperating devices that addresses the varied issues that arise in dynamic environments.

Keywords: Mobile Cloud Computing; Code Partitioning and Offloading; Intelligent Agents; Multi-hop Ad-hoc Mesh Network.

---

* Corresponding author.

1. Introduction

Interoperability in computer systems has turned out to be a feature of paramount importance, especially due to their mutation to distributed systems. Cloud computing has expanded the space and capacity of these distributed systems by providing advanced services and high performance [31]. Considering the high penetration and impact of the mobile phone technology in the human's day-to-day activities, it can be observed that there is a continually growing need for high processing applications to be used through the phone. Consequently, majority of software systems on the mobile phone are distributed, permitting user access to a large plethora of applications, despite the physical limitations in resources of the phone, hence the need for Mobile Cloud Computing [31]. Nevertheless, the mobile phone still faces problems of resource scarceness, finite energy and low connectivity. Considering the trends in Mobile phone architecture, these problems are unlikely to be solved in the future because they are inherent to mobility [21,34,35]. The concept of offloading [36,37] not only data but also computation (also called Cyber Foraging) from the mobile device to the cloud is therefore used to alleviate these inherent problems by using resource providers other than the mobile device itself, to host the execution. However, offloading computation and providing external storage services require Internet connections which may not always be available. Although Internet connectivity is almost ubiquitous, there are many circumstances in which using infrastructureless communication is better than an IEEE 802.11 hotspot or cellular communication, as the latter may be either infeasible, inefficient, or costly. For instance, in remote areas or in natural disaster cases [41,42], there is no infrastructure available. Also in a social insurgence, in which the infrastructure cannot be trusted, in low latency gaming [43,44,47] or in sharing files with colleagues, such infrastructureless technologies can be used. An additional application scenario could be group communication in mass events, like conferences or concerts [44], in which infrastructure may be unable to support all communication demands. Due to these reasons, Mobile Ad-hoc Networks (MANETs) were identified as a major emergent technology at the "Internet on the Move" workshop [44]. Thus, the challenge remains of how mobile devices can rely on their collaborative efforts to provide and to access high computational services. As a result, amidst the Mobile Cloud Computing paradigms, there is a positive prospect for Transient Clouds. A Transient Cloud is a computing platform that allows nearby mobile devices to form a SmartPhone Ad-hoc Network (SPAN) [6] and collaboratively provide various capabilities as cloud services to each of the devices in need [9]. Transient Clouds utilize the collective capabilities of the mobile devices present in a given neighbourhood, capabilities that cannot be provided by traditional clouds in case of lack of internet infrastructure. Transient Clouds are suitable in temporal scenarios in which the cloud is created on-the-fly only by the devices present in an environment and it would change or disappear as the devices roam about or leave the network. A multi-agents system is another distributed computing paradigm that has agents, originating from various devices, able to interact with each other with an intelligent behaviour. Multi-agents systems are often used to solve problems using a decentralized approach, where the agents contribute to the solution by cooperating with each other [19]. Although Transient Clouds and Multi-Agents Systems are independent and different, integrating their use can provide so many advantages. The need for this integration and its benefits can be seen especially when operating in a local mesh network of mobile devices with a discontinuous and unstable connectivity among them, even as it is evident that this type of network is the one that fits the most the scenarios cited above. In this paper, we present a framework that detailly elaborates the concepts related to this integration and these concepts are

expatiated in subsequent sections.

2. Review of Related Work

Computation mobility or cyber foraging [28] is a capability that is critical to the success of mobile cloud computing (MCC). It refers to the ability of the infrastructure to seamlessly migrate a computation from one node to another node. It is desired in cases where computing devices are available in every size and shape, and where sharing of computing resources can be beneficial. In the context of Transient Clouds (TC), smartphones and tablets take advantage of computation mobility. Transient Cloud Computing attempts to alleviate restrictions on smart phone devices (e.g. their computational power, battery life, memory capacity, etc.) by offloading all or some part of the computation on a smartphone to a neighbouring smartphone in the SPAN. In their survey work of the most recent mobile cloud computing systems, Niroshinie and his colleagues [31] demonstrate that the trends in MCC favour Virtual Machine (VM) migration and Mobile code over the conventional Client–Server Communication system; even more when considering the ad-hoc nature of mobile systems. Furthermore, the continuous on-going interaction and communication between the client and server may lead to network congestion. However, in cases where the mobile device user is within the range of a surrogate device for a few minutes, using VM migration may prove to be too heavyweight, as is pointed out in [5] which uses mobile agents in light of its suitability in a dynamic mobile environment. A number of computation mobility solutions have been proposed for MCC like MAUI [29], CloneCloud [30], ThinkAir [26], Scavenger [5] and MobiCloud [25], among others. They offload computation units from a mobile device to a remote server in the public cloud to achieve various performance properties. The above works, however, rely on a functioning Internet connection to perform the offloading – something that may be not be possible in many scenarios. Mobile Clouds, on the other hand, do not rely on public clouds because the computation is performed on the mobile devices. SATIN [31] is a Peer-to-peer system for mobile self-organization, but it is based on component model systems representing systems made up of interoperable local components rather than offloading jobs to local mobile resources. Our work focuses primarily on this latter type. In their paper on mClouds [33], Miluzzo and his colleagues discuss the theory behind the concept as well as what possible incentives might be for getting users to connect their devices to the cloud but do not give details about how the cloud services are assigned to devices. Another work related to Transient Clouds is the OSGi system [45]. The OSGi system features remote installation and execution of code on other devices; the difference is that Transient Clouds do not need to install any code on the devices. It simply sends pre-compiled code to another device and has it execute only the exact computation unit, without any installation. Additional works in the area include Apache Hadoop [46]. Hadoop is a system for distributing computation across multiple machines. Hadoop, however, is designed for a well-known network of servers and server racks, while our framework deals with a heterogeneous set of mobile devices with individual devices leaving and joining the network at various intervals. Priyank and his colleagues [23] propose a trust model based on security agents, which are simple agents that provide security at the virtual machine and the entry point of the network cloud. Okba and his colleagues [24] present an agent based approach designed for the execution of a service (SaaS) in a cloud. It does not consider mobile devices but instead full-fledged computers. It defines a set of components (static and mobile agents) and functional modules described in terms of their behaviour and interfaces, and how these components interact in order to accomplish all the tasks correctly in the system. However, using a SaaS

architecture supposes that the service it needs is available on the other device, therefore there is no code offloading in this case. Scavenger [5] is another framework that employs cyber-foraging using WiFi for connectivity, and uses a mobile code approach with agents to partition and distribute jobs. It also introduces a scheduler for cost assessment based on the speed of the surrogate server. Using its framework, it is possible for a mobile device to offload to one or more surrogates and its tests show that running the application on multiple surrogates in parallel is more efficient in terms of performance. However, it does not discuss fault tolerance mechanisms and since its method is strictly about offloading on surrogates and not sharing, it is not really dynamic. Also its surrogates are all desktops and it is unclear if Scavenger is too heavy to run on mobile phones. Hwirim Byun and his colleagues [8] presented a Mobile Agent Oriented Service framework for offloading on Mobile Cloud Computing. They use a Client Mobile Agent to encapsulate and offload the code on the mobile device to the server side. Their server side is constituted of a Mobile Agent Server and several surrogate servers that use a Server Mobile Agent to manage the execution of the offloaded and to send back the results. In this structure, the servers used are not smartphones but full computers; also the Mobile Agent Server is a weak link as its failure may disrupt the functioning of the structure. Qingfeng Liu and his colleagues [38] elaborated a Universal Mobile Service Cell (UMSC) based framework which is a unique mobile agent based optimization solution for MCC. The proposed architecture is composed of mobile hosts, UMSC, local cloud unit and remote cloud unit. However, this cloud unit again are full computers and the framework uses UMSC for the offloading of entire application to remote cloud unit. Pelin Angin and his colleagues [13] even created a working prototype of an Agent-based Optimization Framework for Mobile-Cloud Computing. They considered code annotation, partitioning and offloading using mobile agents, tested on the Sudoku and NQueens games' source codes, but they offload to the Amazon EC2 cloud server. Terry Penner and his colleagues [9,10] designed a framework for Assignment and Collaborative Execution of Tasks on Mobile Devices in a Transient Cloud. This cloud utilizes the collective capabilities of the devices present, along with their social and context awareness that cannot be provided efficiently by the traditional clouds. They present a modified algorithm of the Hungarian method for assigning tasks to devices in order to achieve various goals (e.g., load balancing, collocating executions, etc...), evaluate the performance of the algorithms through simulation and provide a real implementation on the Android platform using the Wi-Fi Direct framework. This work has the following limitations:

- Firstly, Wi-Fi Direct is a technology that allows devices to create an ad-hoc network and connect to each other using standard Java sockets. In Wi-Fi Direct, only one device (called the Group Owner) acts as a router, and all of the other peer devices that connect to it create a single hop network. Also, any device in a group can only be a client in another group. This limits the possibility of expansion of the network and the size of the network to be a maximum of one hop from the Group Owner.
- Secondly, because of the fact that every device must connect to the Group Owner, a significant disadvantage is that if the Group Owner leaves, the group is torn down and a new group must be established from scratch. This make Wi-Fi Direct unsuitable as a basis for multi-hop networking for moving nodes.
- Thirdly, the group owner is the hub device of the network that is responsible for maintaining the network's state and it needs to have a stable connection with the client devices in the network to route tasks (code) and data.

Therefore, their Task assignment and execution algorithm may hardly work on a multi-hop ad-hoc mesh network with a very unstable connectivity between moving devices, which is normally the typical network for a realistic Transient Cloud. Meanwhile, it is crucial to consider this network topology and its characteristics to produce an application framework which responds to the needs and the features of the TC structure, and the algorithms for task assignment and execution should be derived to align with the framework in question. For this reason, in this paper, we design a framework suitable for this setting with the help of mobile agents.

2.1. Definition of Agents

In the literature, there is no standard definition for an agent. There are several similar definitions, but they vary on the base of the type of applications for which the agent is designed. In this context, Ferber [4] proposes that an agent a physical or virtual entity, which:

- has its own resources;
- is able to interact within an environment;
- is able to perceive its environment (but to a limited extent);
- is powered by a set of patterns (individual objectives and satisfaction/survival functions) that seeks to optimize;
- can communicate directly with other agents;
- provides services and owns capabilities;
- may eventually breed.

This definition raises four essential properties of an agent: autonomy, independence, flexibility, all in view of the structure of its environment and the changes occurring in it. Actually, there are two type of agents. Stationary agents reside on a host, and communicates with its environment using conventional techniques such as the remote procedure calls (RPC) or the notification messages. However, when they need to interact with other agents on remote machines, they are obliged to use communication protocols based on client/server model. Mobile agents are able to communicate with other agents, and roam about freely in the network, while optimizing their itineraries and deciding on the tasks to be executed.

2.2. Making Use of Agents: The Rationale

Mobile Cloud Computing (MCC) Systems are usually based on a Client/Server architecture where the mobile device is the client and request services from the server machine through a network and gets a response. However, in a Transient Cloud, we want to be able to consider every device as a potential server at a time and client at another time. Therefore, in this project, we are interested in harnessing the resources of other mobile phones of the vicinity constituted into an infrastructureless mesh network (that is using only the network cards of the smartphones present). They will be constituting a dynamic local cloud, dynamic because the devices present in the cloud continuously change as the users move, and will be serving as resource providers. This approach supports the user mobility and recognizes the potential of mobile clouds to do sensing as well. After more than a decade of study, mobile agents have gotten a lot of breakthroughs in many key technologies, but

they have suffered from the fact that they could not find an appropriate platform in a large scale network to fully exploit their potential and achieve the expected advantages. The Transient Cloud technology may be one of the best platforms to provide a good chance for the mobile agents to display their aptitudes, especially when dealing with mesh network with multi-hop capability as it is the case of our work.

3. Qualities and Advantages of Mobile Agents in the Transient Cloud Model

As a matter of fact, there are several advantages that motivate and promote the use of the mobile agent technology over the traditional client/server model, in the building of distributed systems in general. A mobile Agent (MA) runs in an environment, using its resources and a set of services (SaaS, PaaS, IaaS of Cloud Computing) provided by the hosting platform. The very first services that are of interest in the case of our work are the processing power and the battery life of the host; that is IaaS. As such, some advantages of using mobile agents, according to [2] are:

— Reducing the network load: communication within the distributed system, which is very substantial to ensure its proper functioning in MCC, usually requires multiple interactions, to accomplish tasks. Meanwhile, mobile agents allow clients to package the conversations and dispatch them to the destination host, in order to perform and deal with interactions locally. This helps reduce the flow of raw data in the network and overcome network latency since the data is processed in its locality rather than transferred over the network.

— Asynchronous and autonomous execution: the proper implantation of mobile devices require expensive network connections, which are continuously open without interruption. This is greatly expensive at the economic and technical levels, and cannot be feasible in a pervasively ad-hoc network, as it is the case of a TC on a SPAN. As a solution for this issue, tasks can be eventually packaged into mobile agents, which can then be dispatched into the network. At this moment, the agents become independent of their owners and can carry out executions asynchronously and autonomously. Later, the mobile device can reconnect and gather results from the agent.

— Dynamic adaptation: mobile agents are able to sense their execution environment, and thus respond autonomously to changes. Besides, agents may work in communities and spread themselves among multiple hosts in the network, in order to maintain the optimal configuration for solving a particular problem. Also when a host is being shut down, the mobile agents are informed to update their itineraries to eliminate the host in question; moreover all agents executing on that host are warned to move to another destination, where they can pursue their execution. It therefore becomes easier to build fault-tolerant and robust systems through the use of mobile agents, that react autonomously and dynamically to inappropriate events and incidents.

A standard MCC system, on a local or public cloud, uses a client/server architecture and therefore has the following limitations:

- It is always the software classified as client that initiates the communication via a service request, while

the server waits passively to be activated and queried;
- It is costly due to the technical nature of the server;
- The server is a weak link given that the entire network is built around it.

Therefore, still in the specificities of this context and in relation to those listed above, the advantages of Mobile Agents over Client/Server are as follows:

- The agents can be configured to search for information about the devices in the cloud in a smarter way;
- Agents create their own knowledge bases that are updated after each search, based on their current needs and on the resources available in the network;
- The communication and cooperation among agents accelerates and facilitates processes;
- The execution of specialized MAs offers flexibility and allows for more robust transactions.

In this context of a Transient Cloud, the concept of mobile agents appears to be an innovative solution. Firstly, they can be used to enhance the stability [39] and improve the performance [15] of the SPAN alongside some routing protocols specifically adapted and/or created for SPANs. Secondly, with mobile agents created on each device in the SPAN, each agent/device can render services (be a server) or request for services (be a Client) at any time. A device having multiple agents can therefore behave as server and a client almost simultaneously. Thirdly, it is worth to mention that the perimeter of a SPAN community in general evolves over time, depending on system failures and the integration of new resources since the network is constituted on-the-fly. In this mesh topology, given a destination, mobile agents can autonomously, among many other programmable actions:

- move between hosts through their execution process;
- exchange information among themselves to monitor the network and if any change happens;
- determine their routes with the help routing protocols;
- change route or host in case of any discrepancy.

4. Constitution of the SPAN for the Transient Cloud

4.1. Network Connectivity in a SPAN

In the context of the network described above with the cloud made up of smartphones essentially, the two main technologies that can be used to provide connectivity in a SPAN are the Bluetooth and the 802.11 (protocols for the implementation of LAN Wi-Fi computer communication). Having that Bluetooth has a shorter range of access area and lesser data transmition rate than Wi-Fi, we would be considering more of solutions that make use of Wi-Fi. Of course, the 802.11 standard [40] provides the infrastructure mode and the infrastructureless (ad-hoc) mode. The latter mode is the one which is of interest to us. Once the means for connectivity among the devices is settled, the next item to address is the routing mechanism.

4.2. Routing Protocols for a SPAN

Routing in a MANET in general typically involves two phases: route discovery and route maintenance. Route Discovery is the means by which a source node S intending to forward a packet to a destination node D,

effectively obtains a route to D. Route Maintenance is the means by which a source S is able to detect, while using a route to node D, that one or more links along the route are unsuccessful. When a broken link is discovered, the source can use another route or can re-invoke Route discovery. MANET routing protocols are usually classified into two categories – proactive and demand-based. With proactive routing, the information on all available paths is continually maintained using periodic updates; so when a packet needs to be sent, routes are known and can be used directly. The proactive method takes little to discover routes but must maintain routing information for unused paths. Examples are the Optimized Link State Routing protocol (OLSR) […] and the Topology Broadcast using Reserve-Path Forwarding protocol (TBRPF) […]. Demand-based routing, rather than maintaining paths between all nodes at all times, invokes a route discovery to procure base on need. Demand-based schemes use less network bandwidth as they avoid sending unnecessary routing information but they typically take longer to discover routes. Examples are the Ad-hoc On-Demand Distance Vector protocol (AODV) […] and the Dynamic Source Routing Protocol (DSR) […]. In the context of implementing MANET over smartphones, therefore having a SPAN, lightweight routing protocols have been proposed in [16]. They refer to those protocols that require minimal memory for routing table, less computing resources and generate less protocol control overhead. They are therefore proactive protocols that do not depend on global network information or hybrid protocols that combine proactive and reactive protocols. Examples are the Better Approach to Mobile Ad-hoc Network protocol (BATMAN) [1,16], the Zone-based Routing Protocols (ZRPs) [16] and the Cluster-based Routing Protocols (CRPs) [16]. Among all these, BATMAN [2] is used more often reason being that it does not need to maintain full path to the destinations. Each node only collects and maintains the information about the best next hops towards all other nodes in the network. Every node collect this information through hello packets broadcast periodically. This makes the protocol suitable for storage constrained devices. Also, since this protocol depends only on hello packets to know the availability of nodes and does not broadcast topology change messages, the control overhead is low. In the literature, a lot of work has been done to upgrade many of these protocols or even create new ones using the latter as basis, again with the help of mobile agents [39,15,18]. There, mobile agents on various nodes are used to roam about the network (See Figure 1), collect information and cooperate to forward data for each other to allow communication over multiple hops between nodes not directly within wireless transmission range of one another. They are equally used for route discovery by continuously tracking the network topology and updating routing tables at all mobile hosts reached. When a route is requested, an agent is sent to discover routes to the destination. These agents analyse the routing tables on the hosts they arrive at and either return a discovered route to the sender or move onto another machine if no route is found. They may do all this while considering the level of congestion on the routes, the utilisation of bandwidth and energy consumption on the devices. And since they are autonomous, they can reduce network load and latency by running remotely. Bindiya Bhatia and his colleagues [18] describe the role of mobile agents in the layered architecture of mobile ad-hoc networks (Data Link Layer, Network Layer, Transport Layer, Application Layer, Cross-Layer).

4.3. Some Existing Proofs of Concept Software Applications for SPANs

The latest amendment of the 802.11 standard [40] provides two different modes that can be used for ad-hoc networking: Independent Basic Service Set (IBSS) mode and 802.11s.

- The IBSS mode of 802.11 is commonly referred to as ad-hoc mode because it does not require any infrastructure to be in place. It can be used as a basis for mesh networking. In this mode, all nodes play similar roles, and any node can communicate directly with any other node within the network, as those nodes are also set to the IBSS mode, share the same Service Set Identifier (SSID) and are within its radio range. The IBSS mode itself, however, does not offer multi-hop capabilities. There is no provision for path discovery and selection, nor for relaying packets to nodes out of the radio range of the sender. In IBSS-based ad-hoc networks, these functions must be accomplished by an additional protocol, like OLSR [16] or BATMAN [1,16], usually at the network layer. Since connectivity is provided mainly at the link layer (optionally with additional network layer support for multi-hop), IP-based applications work without any modification in an IBSS-based network.
- The 802.11s has been introduced more recently with multi-hop mesh networking now incorporated in the standard. 802.11s defines the Mesh Basic Service Set (MBSS) that provides a wireless Distribution System (DS) based on meshing at the link layer. 802.11s defines a standard protocol, Hybrid Wireless Mesh Protocol (HWMP), though others can be used as long as all stations in the mesh agree. HWMP combines reactive routing derived from Ad hoc On-demand Distance Vector (AODV) with root-based proactive routing for communication with the outside. Unfortunately, majority of the current wireless chipset drivers and the operating systems on smartphones do not yet support this technology and need to be customized [11].

For the partitions of a software on a smartphone to be disseminated on a SPAN, it would need to implement the use any of the protocols cited above and set-up an IBSS-based network to connect with other devices; this before executing its code partitions. This implies altering the source code of all network applications to this new adaptation. However, the software in question may use a multi-hop ad hoc networking application that acts as a middleware between it and the OS kernel, does all the necessary network set-up, and needing little or no change in its code of the requesting software. Also, they are to be installed directly on the smartphone as any other software. Some of these solutions are offered in [11,12]. Reference [11] for example, presents AdHocDroid which is an Android application that makes the necessary changes in the device to effortlessly create a MANET. The application sets up the IBSS network, enabling ad-hoc mode on the wireless card, offers the possibility to choose the network name, and configures the IP address, network mask and gateway for the device though all the parameters have default values. The application also allows an easy way to import and run different routing protocols, and using tools to monitor and evaluate the state of the network. It is implemented for the Android OS and is made publicly available to the community by the authors.

5. Proposed Framework

5.1. Architecture of a Mobile Agent based Transient Cloud on an Ad-hoc Mesh Network

The mobile agent based Transient Cloud is feasible because most agent systems are based on or support Java such as Aglets [20], JADE (JAVA Agent Development Framework) [27]. They support the development of agents with the ability to transport them from one system to another. Even though mobile devices nowadays have different Operating Systems (OS) (Android, Windows, ios, etc…), since with Java, one can “write once and run anywhere”, the mobile agent can run on the Java Virtual Machine (JVM) installed on the OS. This

mechanism will help to realize portability and interoperability between heterogeneous devices. In this work, the following set-up is therefore proposed. A JVM and a Mobile Agent Zone (MAZ) are preinstalled on every mobile device susceptible to participate in the Transient Cloud Space (TCS). If the device in question is a Java Phone (eg: with the Android OS), then there is no need to install the JVM. This process can be done before hand on every device before its sale like the installation of any other key software, or based on the willingness of a user to contribute to the cloud setup with his/her mobile device in exchange of an incentive. The aspect on incentive will be studied in a subsequent work. The Mobile Agent (MA) is created by and runs in a MAZ provided by the OS on a device. It can move from one MAZ to the other among the different devices though they are heterogeneous. The MAZ on every device actually acts as a task manager. At the same time, interoperability and communication is realized among agents through standards such as MASIF (Mobile Agent System Interoperability Facilities Specification) [22] and/or FIPA (Foundation for Intelligent Physical Agents) [3]. A MAZ on a device holds the information of the mobile agents residing on it and the level of resources available on it. It frequently renews its status and publishes its availability through data packets to the other devices in the network. Every MAZ has the information of directly connected devices (best next one hop as the BATMAN [2] protocol provides). The maximum radius of the TCS depends on the wireless network technology used for the interconnection of the devices. We consider the IBSS mode of the 802.11 ad-hoc standard. Based on various criteria like resources needed, network connectivity and distance, the MA may autonomously and progressively determines its trajectory (hosts for execution) as it is moving, one device (hop) after the other till the end of the execution of its task. This node allocation strategy is expounded upon in a subsequent work. The MAZ registers the information of every newly created MA in the initial section. Each MA of an application, is tagged according to the application and device it originates from, and is sent to the MAZ of the next chosen device. When the MA leaves the MAZ, its information is kept in the pending section till its return. When the MA returns, the MAZ collects the resulting information, unregisters the MA, and destroys it. The MAZ is also responsible for the awaiting queue (where the MA waits for its turn of execution) and monitors the agents on it. Furthermore, many services including resource indexing, authentication, security, billing, disaster recovery and fault tolerance are provided by the MAZ. The structure of the MAZ is given in Figure 3.

At the user's end, a task is encapsulated in the mobile agent. In order to offer better compatibility among the different OSs, all mobile agents will have to have the same structure (See Figure 2):

- The Type: usually a flag that permit to determine if the packet entering the MAZ is a MA or no.
- Application Code: The offloaded code that the MA carries towards execution.
- The MA Code: a sequence of instructions to be executed, in this case a Java code, which defines the static behaviour of the mobile agent. It also has the Execution Context that reflects the current execution state of the mobile agent (registers values, execution stack).
- The Requirements: the needs of the offloaded code partition, in terms of data and resources. There are two types:

— Transferable Resources: they represent the attributes values of the agent, which provide him with a global state.
— Non-Transferable Resources: they constitute the execution environment provided by the system (open

files, sockets, connections, registers, etc.) and the physical materials used by the agent (printer, monitor, etc.).

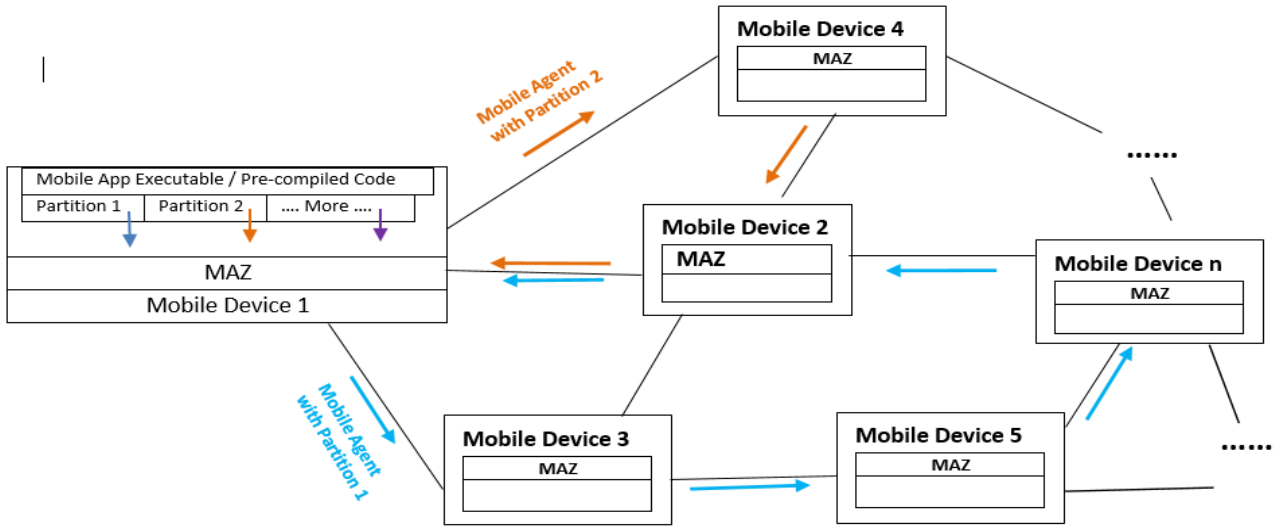

Figure 1: Mobile Agent based Transient Cloud on a Multi-hop Ad-hoc Mesh Network

Type	Requirements	MA code	Application Code
- Flag: Is it a MA or no?	Demands for hardware, software and any other types of resources (transferable and non-transferable)	- Index of the MA - Device/Application of Origin - Status (done or not done)	- tasks to be executed

Figure 2: Structure of the Mobile Agent (MA)

Initial Section	Pending Section	Remote Resource Section	Execution Section	Awaiting Queue

Figure 3: Structure of a Mobile Agent Zone (MAZ)

The MA will check the Remote resource section of the MAZ to find the available resources on other devices in the TCS, and will run an algorithm to choose the most suitable device as next hop for its execution. Then, the MA moves to the execution section of the targeted MAZ/device in the TCS. Finally, the MA will execute on the host till the completion of its mission and return directly to the mobile device MAZ where it originated from. If the MA does not complete its mission due to lack of resources, it runs the algorithm again to choose the next device for the continuation.

At the moment of transitioning, depending on its execution level, the MA will execute one of the two types of migrations [14]:

- Strong migration in case the agent did not complete its task, so it needs to migrate carrying the code and exact execution state. The agent is suspended, marshalled, transmitted, unmarshalled and then restarted at the exact position where it was previously suspended on the destination node without loss of data or execution state.
- Weak migration in case the agent has completed a task and is transporting the result.

6. Application Partitioning and Computational Granularity

Application partitioning is meant as the splitting of the resource-intensive application into computational tasks (parts) with the intention of executing them on the cloud devices having considerable amount of resources. Generally, two types of partitioning are considered: static partitioning and dynamic partitioning. The static partitioning implicates separation of computational intensive components at compile time statically [11] and results into fixed number of partitions. It doesn't consider varying load of CPU, network parameters and can't utilize the elastic property of cloud resources (expand one task to several devices). So, dynamic partitioning seems to be the best option for partitioning. Dynamic partitioning is a technique which partitions an application at run time taking into account varying CPU load on the mobile devices in the TCS and network parameters and utilizes elastic resources of the cloud. Also, two costs are considered for partitioning: computation cost and communication cost. The computation cost refers to the time and the level of resources required for processing the components on both the mobile device and the cloud. The communication cost constitutes the cost required for transferring the data and control information between the mobile device and the cloud. There is need to decide the level of granularity at which the code should be partitioned and offloaded. The possible granularity variations that are usually supported for a single agent (in a descending order) are: thread, process, method, object, application-level component, entire application [14]. The objective for this part therefore is to get the best possible dynamic partitioning strategy that will give the least amount of resource consumption and use the fewest possible number of devices in the cloud, all this while having a short total execution and transfer time. In the same sense, the partitioning of the code should be programmatically automated and the level of granularity should change dynamically from the lowest to the highest level possible, as the need arises. Also, the MA may hold about of 200 lines of application code to remain as lightweight as possible. For the framework here proposed in the dynamic approach, each mobile application consists of a set of tasks (code partitions) that are offloadable to the TCS for execution. In order to have an agent execute a specific task, the task should be implemented and annotated using the relevant API designed for that purpose [13]. In addition, the mobile application has a set of native components that are always executed on the device due to constraints such as accessing native sensors of the device or providing the user interface of the application. These suggest that the source code of the application is analysed and profiled during the development phase of the application, and the offloadable tasks are thereby determined and annotated. These type of tasks are usually (with some exceptions) independent from others and/or can run parallelly. Considering the fact that the devices joining the TCS may use different operating systems (Android, Windows, etc.), the tasks to be offloaded should be developed with an interpreted language alongside the API in order to be executable on all types of devices. When an application

wants to run and realizes through the Operating System that the resources are not enough on the mobile device, it contacts the MAZ, through the API, to supply the number of hosts available and the interval range of resources (amount of CPU power, Battery life, etc) available on them. Therefore, the API should be able to:

- Tag as "Agent" the sections of code that are offloadable, so that in case a code partition is sent to the MAZ, the later should distinguish it among all the other sorts of packets it receives and package it into one or more agent (s);
- Examine each annotated partition to select **the ones whose needs for resources fall within the given range**;
- Send to the MAZ each of the so selected code partitions.

In this scenario, the number of code partitions submitted to the MAZ is ideally less than or equal to the number of available hosts in the TCS, but it could be more. This implies that some offloadable tasks may finally not be offloaded, but executed instead locally. Also, since an annotated task can have several nested annotated tasks, the agent built from it can easily breed into smaller agents using those imbrications. Thus, the number agents emanating from one application is not fixed, but it changes dynamically. As an independent software, the MAZ may determine, during its installation on the device, the amount of space that it will require to host a certain number of agents and run its activities. The MAZ also acts as a security barrier to protects the agents and control the data exchanged between the device host and the agents. As a wrap-up, the proposed mechanism is as follows:

Table 1: Stepwise Summary

Step Number	Activities carried out
1)	When an application does not have enough resources to run on a device, the application will be partitioned at the level indicated in the code and those snippets of code are sent to the MAZ with the tag "agent".
2)	The MAZ reads the head of each package received to judge if it is a MA or other kinds of data packages. Once the MAZ receives the code snippets, it creates agents and encapsulates each snippet. This marshalling process of the agents follows the structure described in Figure 2. Then, it registers them in the Initial section of the device's MAZ.
3)	Then, the MA matches its requirements with the resource index on the Remote resource section of the MAZ, to decide on which device to go to.
4)	Once the next MAZ receives the MA in its execution section, it activates the MA, unmarshalls it and executes the task included in it.
5)	The MA monitors the execution of the task and the status of the resources in the MAZ, therefore deciding whether to leave the MAZ to another one, or breaks into new smaller MAs and send them to other MAZs in the TCS to accomplish the task.
6)	Mobile agents can negotiate and collaborate with each other to exchange data and realize the inter-operability among the different devices.

All the activities performed by the MAZ are according to the local management regulations instituted by the objectives of the MAZ and the goals of the agents in order to move agents around, when resources are scarce on the originating device.

The MAZ on each device should be able to run the following activities:

- Gather information about the resources available on the devices in the TCS and present it to the MA that arrives in;
- Keep track of the MAs that come, also of when they leave;
- Publish the availability of the host device when it comes into the network and the resources that are obtainable on it;
- Update the list of available devices and their resources when a new device comes in;
- Controls the life cycle of agents by creating, suspending, resuming and killing them.

Furthermore, the MAZ should have a security module to verify the request and the degree of safety of the agent (data integrity and authentication).

The MA does the following:

- Check out the list of devices available on the network when it is stuck in a location and/or need to move;
- Determine the next node of its itinerary based on the list of available devices that the MAZ currently has, up until its return to the requesting device;
- Communicate and exchange data with other agents;
- Split itself into smaller agents depending on the context.

Once the agent is transported to the destination hosts, the agent can go ahead and execute to completion even in the case of a disconnection in the network; and when the connection is back, the agent can travel to the next host or return home. Thus, the overall mechanism can provide an almost infinite resource pool for the use and realize a high scalability of the cloud computing resources. Nevertheless, there is a need to minimize the amount of messages and agents moving through the network. This Figure presents the architectural stack that suits the scenario described of what happens on the user's mobile device found in a TCS.

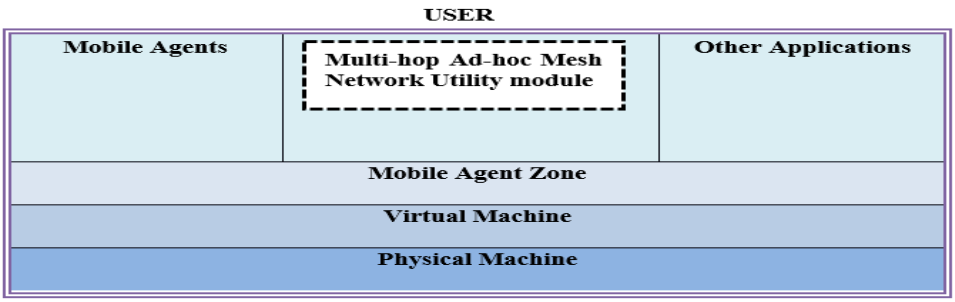

Figure 5: A Smartphone's Architectural Stack in a TCS based on Mobile Agents

It can be supposed that a device is triggered to join the network either automatically when it reaches the TCS or manually by the user based on an incentive. If users are to be persuaded to collaborate and share their resources with others, there needs to be a motivation either through monetary or social incentives to do so [7,17]. An interesting method is using common goals, but in the absence of common activities this will not prevail. The

user may also be given the possibility to use, at a later time, equal amount of resources that he has given out, plus an interest rate. The aspect of incentive will be evaluated in another work.

7. Conclusion and limitations

In this paper, we have designed a framework for efficient cyber foraging on a Transient Cloud built on a multi-hop ad-hoc mesh network, making use of mobile agents. Considering the atypical characteristics of such a network, we have considered the available technologies and protocols that could respond best to its need, and then based our thoughts on this knowledge in other to develop a framework that fits the details. As elaborated above, compared to several other frameworks for code partitioning and offloading as the ones presented in this work, the proposed architecture addresses best the various issues that arise in such a dynamic environment. However, this framework has some limitations like offering a trust model for securing data and a payment model for the users in the Transient Cloud who lend the resources of their devices and ensuring the security, the integrity and the privacy of data for all users. It is will be beneficial to carry out a technology acceptance study to evaluate the level of interest of potential users of this technology.

8. Further Work

To assess this framework, we have three main action points. Firstly, we will have to design a task assignment and execution algorithm that suits our proposed framework. The algorithm would take into consideration the fact that the agents are aware of the context on the network and they can gain experience from user's behaviour (device's resources usage, mobility speed and direction, distance between devices etc…) by some advanced artificial intelligence means. Secondly, we will have to develop a prototype of the system with a mobile agent simulator. Lastly, we will consider implementing and testing this framework on a real life application. Supposing that an application is partitioned into N tasks on a device, we want to gage the amount of time and resources the tasks take to be completed, and also evaluate the performance and stability of the network, when making use of the other devices in the Transient Cloud with the help of mobile agents on the base of the framework described above. Then, we will compare the results to the case of executing on just one device and executing in a Transient cloud without agents. A simulator software we could use is Repast Symphony. It is a richly interactive, easy to learn, expert focused, agent-based modelling and simulation system that is designed for use on workstations, computing clusters and super computers. It is open source and mainly java-based.

References

[1]. Dewiani, M Baharuddin, M F B Gufran, S. Panggalo and Wardi, "Performance of Routing Protocol OLSR and BATMAN in Multi-hop and Mesh Ad Hoc Network on Raspberry Pi", Conf. Series: Materials Science and Engineering (2020)

[2]. Benjamin Sliwa, Christian Wietfeld and Stefan Falten, "Performance Evaluation and Optimization of B.A.T.M.A.N. V Routing for Aerial and Ground-based Mobile Ad-hoc Networks", 978-1-7281-1217-6/19 IEEE (2019).

[3]. Foundation for Intelligent Physical Agents. "FIPA Agent Management Support for Mobility

Specification", document number dc00087c. Technical report, Geneva, Switzerland, May (2002).

[4]. J. Ferber. "Multi-agent systems: an introduction to distributed artificial intelligence", volume 1. Addison-Wesley Reading. (1999).

[5]. M. Kristensen, Scavenger: transparent development of efficient cyber foraging applications, in: Proceedings of the IEEE International Conference on Pervasive Computing and Communications, PerCom (2015).

[6]. Josh "m0nk" Thomas, Jeff "Stoker" Robble, Off Grid communications with Android: Meshing the mobile world, The MITRE Corporation Bedford, MA USA (2015).

[7]. Abdullah Gani, Ejaz Ahmed, Ibrar Yaqoob, Muhammad Imran, Salimah Mokhtar and Sghaier Guizani, "Mobile ad hoc cloud: A survey", Wireless Communications and Mobile Computing (2016).

[8]. Boo-Kwang Park, HwiRim Byun and Young-Sik Jeong, "Mobile Agent Oriented Service for Offloading on Mobile Cloud Computing", Advances in Computer Science and Ubiquitous Computing, Springer Nature Singapore (2017).

[9]. Alison Johnson, Brandon Van Slyke, Mina Guirguis, Qijun Gu, Terry Penner, "Transient Clouds: Assignment and Collaborative Execution of Tasks on Mobile Devices", Globecom 2014 – Symposium on Selected Areas in Communications: GC14 SAC Internet of Things, (2014).

[10]. Agustin Rivera-Longoria, Alison Johnson, Brandon Van Slyke, Mina Guiguis, Lavanya Tammineni, Qijun Gu, Terry Penner, Thomas Langford, "Assignment and collaborative execution of tasks on transient clouds", Institut Mines -Telecom and Springer-Verlag France SAS, 27, (2017).

[11]. Ana Aguiar, Eduardo Soares, Pedro Brandão, Rui Prior, "Experimentation with MANETs of Smartphones", arXiv:1702.04249v1 [cs.NI], (2017).

[12]. Paul Baskett, Tiancheng Zhuang, and Yi Shang, "Managing Ad Hoc Networks of Smartphones", International Journal of Information and Education Technology, Vol. 3, No. 5, (2013).

[13]. Bharat Bhargava, Pelin Angin, "An Agent-based Optimization Framework for Mobile-Cloud Computing", Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, volume: 4, number: 2, pp. 1-17, (2013).

[14]. Ichiro Satoh, "Book Chapter on Mobile Agents", National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan, (2014).

[15]. Manoj Jhuria, Shailendra Singh, "Improve Perfomance DSR Protocol by Application of Mobile Agent", Fourth International Conference on Communication Systems and Network Technologies, (2014).

[16]. Md Shahzamal, "Lightweight Mobile Ad-Hoc Routing Protocols for Smartphones", Macquarie University, Sydney, Australia, April (2018).

[17]. Abdullah Gani, Ejaz Ahmed, Ibrar Yaqoob, Muhammad Imran, Salimah Mokhtar and Sghaier Guizani, "Mobile ad hoc cloud: A survey", Wireless Communications and Mobile Computing, Wiley Online Library, (2016).

[18]. Bindiya Bhatia, M. K. Somi, Parul Tomar, "Role of Mobile Agents in the Layered architecture of Mobile Ad-Hoc Networks", International Journal of Computer Networks an Information Security, Volume 11, Page 37 – 45, (2015).

[19]. D. B. Lange and M. Oshima. "Seven good reasons for mobile agents". Commun. ACM, ACM, vol. 42,

88-89. (1999).

[20]. H. Tai and K. Kosaka. "The Aglets project". Commun. ACM, 42, 100- 101. (1999).

[21]. M. Satyanarayanan, Mobile computing, Computer 26 81–82 (1993).

[22]. C. Tham, D. Lange, J. White, M. Oshima, S. Virdhagriswaran, and K. Ono. "MASIF: The OMG Mobile Agent System Interoperability Facility". Personal and Ubiquitous Computing, 2(2): 117-129, June (1998).

[23]. Mukul Manmohan Meghwal, Priyank Singh Hada, Ranjita Singh, Security Agents: A Mobile Agent based Trust Model for Cloud Computing, International Journal of Computer Applications (0975 – 8887), Volume 36– No.12 (2011).

[24]. Alwesabi Ali , Almutewekel Abdullah, Okba Kazar, Implementation of Cloud Computing Approach Based on Mobile Agents, International Journal of Computer and Information Technology (ISSN: 2279 – 0764), Volume 02– Issue 06 (2013).

[25]. Dijiang Huang, Jim Luo, Myong Kang, and Xinwen Zhang. Mobicloud: building secure cloud framework for mobile computing and communication. In Service Oriented System Engineering (SOSE), 2010 Fifth IEEE International Symposium on, pages 27–34. IEEE (2010).

[26]. Andrius Aucinas, Mortier Sokol Kosta, Pan Hui and Xinwen Zhang, Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading. In INFOCOM, 2012 Proceedings IEEE, pages 945–953. IEEE (2012).

[27]. A. Poggi, F. Bellifemine and G. Rimassa. "JADE: a FIPA2000 compliant agent development environment" .AGENTS ’01: Proceedings of the fifth international conference on Autonomous agents, 216-217. ACM (2001).

[28]. M. Sharifi, O. Kashefi and S. Kafaie, “A survey and taxonomy of cyber foraging of mobile devices”, IEEE Communications Surveys Tutorials, 14(4):1231–1243 (2012).

[29]. Aruna Balasubramanian, Alec Wolman, Dae-ki Cho, Eduardo Cuervo, Paramvir Bahl, Ranveer Chandra, and Stefan Saroiu, “Maui: making smartphones last longer with code offload”, In Proceedings of the 8th international conference on Mobile systems, applications, and services, pages 49–62. ACM (2010).

[30]. Ashwin Patti, Byung-Gon Chun, Mayur Naik, Petros Maniatis, and Sunghwan Ihm, Clonecloud: elastic execution between mobile device and cloud. In Proceedings of the sixth conference on Computer systems, pages 301–314. ACM, (2011).

[31]. Niroshinie Fernando, Seng W. Loke, Wenny Rahayu, Mobile cloud computing: A survey, Future Generation Computer Systems, Elsevier, online 6 June (2012).

[32]. C. Mascolo, S. Zachariadis, W. Emmerich, Satin: a component model for mobile self organisation, in: R. Meersman, Z. Tari (Eds.), On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE, in: Lecture Notes in Computer Science, vol. 3191, Springer, Berlin, Heidelberg, pp. 1303–1311 (2004).

[33]. E. Miluzzo, R. Caceres, and Y. Chen, “Vision: mClouds - Computing On Clouds of Mobile Devices,” in Proceedings of the third ACM workshop on Mobile cloud computing and services, UK (2012).

[34]. J. Flinn, M. Satyanarayanan, S. Park, Balancing performance, energy, and quality in pervasive computing, in: Proceedings of the 22nd International Conference on Distributed Computing Systems,

IEEE, pp. 217–226 (2002).

[35]. M. Satyanarayanan, R. Balan, S. Park, T. Okoshi, "Tactics-based remote execution for mobile computing," in: Proceedings of the 1st International Conference on Mobile Systems, Applications and Services, ACM, pp. 273–286 (2003).

[36]. H. Bal, N. Palmer, R. Kemp, T. Kielmann, Cuckoo: a computation offloading framework for smartphones, in: Proceedings of The Second International Conference on Mobile Computing, Applications, and Services, MobiCASE, (2010).

[37]. A. Patti, B.-G. Chun, M. Naik, P. Maniatis, S. Ihm, Clonecloud: elastic execution between mobile device and cloud, in: Proceedings of the Sixth Conference on Computer Systems, EuroSys'11, ACM, New York, NY, USA, pp. 301–314 (2011).

[38]. Jicheng Hu, Qingfeng Liu, Xie Jian, "An Optimized Solution for Mobile Environment Using Mobile Cloud Computing", Google Scholar, Wuhan 430079,China, the National Innovative Project (2009).

[39]. C. P. Chang, S. C. Wang, K.Q. Yan, Y. P. Lin, "Enhance the Stability of MANET by Using Mobile Agent", Proceedings of the First International Conference on Scalable Information Systems, Hong Kong (2006).

[40]. IEEE Association, "IEEE Standard for Information technology– Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," IEEE Std 802.11-2012, pp. 1–2793 (2012).

[41]. C. Poellabauer and P. Mitra, "Emergency response in smartphone-based Mobile Ad-Hoc Networks," in IEEE Int. Conference on Communications (ICC). IEEE, pp. 6091–6095, (2012).

[42]. C.-M. Lin, K. Boussetta, J.-R. Jiang, M. Abdallah, T. Y. Huang, and W. T. Ooi, "SYMA: A Synchronous Multihop Architecture for Wireless Ad Hoc Multiplayer Games," in IEEE 17th Int. Conference on Parallel and Distributed Systems, (2011).

[43]. A. Le, A. Markopoulou, C. Fragouli, L. Keller, "MicroPlay: a networking framework for local multiplayer games," in Proc. of the 1st ACM Int. Workshop on Mobile gaming-MobileGames, New York, USA, (2012).

[44]. H. Scholten, P. J. Havinga, O. Turkes, "BLESSED with Opportunistic Beacons: A Lightweight Data Dissemination Model for Smart Mobile Ad-Hoc Networks," in Proc. of the 10th ACM MobiCom Workshop on Challenged Networks (CHANTS '15). ACM Press, (2015).

[45]. A. Sathiaseelan and J. Crowcroft, "Internet on the move: Challenges and solutions," SIGCOMM CCR, vol. 43, no. 1, (2012).

[46]. OSGi Alliance Staff, "OSGi Alliance," http://www.osgi.org/Main/ HomePage, (2013).

[47]. The Apache Software Foundation Staff, "Apache Hadoop!" http:// hadoop.apache.org/, (2013).

[48]. D. Srikrishna and R. Krishnamoorthy, "SocialMesh: Can networks of meshed smartphones ensure public access to twitter during an attack?" IEEE Communications Magazine, vol. 50, no. 6, pp. 99–105, (2012).