A Game-Theoretic Based QoS-Aware Capacity Management for Real-time EdgeIoT Applications

Suleiman Onimisi Aliyu, Feng Chen and Ying He
Software Technology Research Laboratory (STRL),
De Montfort University,
Leicester, UK
soaliyu@hotmail.com, fengchen@dmu.ac.uk,
ying.he@dmu.ac.uk

Hongji Yang
Center for Creative computing,
Bath-Spa University,
Bath, UK
h.yang@bathspa.ac.uk

***Abstract*— More and more real-time IoT applications such as smart cities or autonomous vehicles require big data analytics with reduced latencies. However, data streams produced from distributed sensing devices may not suffice to be processed traditionally in the remote cloud due to: (i) longer Wide area network (WAN) latencies and (ii) limited resources held by a single Cloud. To solve this problem, a novel Software-defined network (SDN) based InterCloud architecture is presented for mobile edge computing environments, known as EdgeIoT. An adaptive resource capacity management approach is proposed to employ a policy-based QoS control framework using principles in coalition games with externalities. To optimise resource capacity policy, the proposed QoS management technique solves, adaptively, a lexicographic ordering bi-criteria coalition structure generation (CSG) problem. It is an onerous task to guarantee in a deterministic way that a real-time EdgeIoT application satisfies low latency requirement specified in service level agreements (SLA). CloudSim 4.0 toolkit is used to simulate an SDN-based InterCloud scenario, and the empirical results suggest that the proposed approach can adapt, from an operational perspective, to ensure low latency QoS for real-time EdgeIoT application instances.**

Keywords*— *Software Defined Networks (SDN), Capacity management, Quality of Service (QoS) control, InterCloud, Edge computing, Internet of Things (IoT), Coalition games with externalities

I. INTRODUCTION

With the Internet of Things (IoT) being a multidisciplinary ecosystem, scenarios demanding real-time big data processing and feedback such as the connected and autonomous vehicles scenarios may increase in demand. As the IoT big data streams are transmitted to the cloud in high volumes and at quick velocity, it becomes necessary, together with the promise of future radio networks (e.g. 5G networks), to design an efficient data processing architecture to investigate valuable information in real time [1]. As is the case with IoT implementation, data is continuously produced and consumed at the edge of the network. It is safe to infer that the number of things at the edge of the network will develop to more than billions in the coming years. Thus, raw data produced by them will be enormous, making conventional cloud computing not efficient enough to handle all these data. Although clouds promise the illusion of infinite resources, nevertheless, the ability for a single cloud's infrastructure to service requests for provisioned resources is limited [2-4]. As such, conventional cloud computing is not efficient enough to handle all these data, and it would be most beneficial (to reduce latency) to process this data as close to the source (i.e. edge of the network) as possible.

Edge computing is borne out of the success of cloud services and a gradual rise in applications related to IoT. Edge computing refers to the enabling technologies allowing computation to be performed at the edge of the network, on downstream data on behalf of cloud services and upstream data on behalf of IoT services [5]. Recently, mobile edge computing (MEC), a new paradigm, was proposed to extend cloud-computing capabilities and services to the edge of the network [6, 7]. MEC-enabled IoT, also called EdgeIoT, can support applications and services with reduced latency and improved Quality of Service (QoS). However, the EdgeIoT appeals for novel architectures and scheduling techniques to address the resource capacity management issue.

A promising solution to the problem of insufficient resource capacity in monolithic cloud systems is to enable cloud interoperability or inter-cooperation [8]. It is inconceivable that each cloud system will have ubiquitous geographical footprints (of data centers) to satisfy the low latency requirements of future real-time IoT applications. Moreover, applications that require copious amounts of computing resources that can only be supplied by a federation of clouds may become increasingly common [9]. Software-defined networks (SDN) refer to an innovative approach for data center network programmability that is, the capacity to initialise, control, change, and manage network behavior dynamically via open interfaces [10]. SDN is a network environment that relies on (i) decoupling the control plane from the data plane, (ii) logically centralised controller and (iii) a standard protocol, such as OpenFlow [11], for communication between the controller and the forwarding elements in the network. SDN heralds unprecedented simplification in network programmability, management and innovation by service providers, and hence, its control model presents itself as a candidate solution to the challenges in network virtualisation.

In this research, a self-adaptive SDN-based InterCloud architecture is presented for real-time EdgeIoT applications.

This paper addresses the problem of 'elastic' capacity scheduling in SDN-based InterCloud architectures, particularly for future MEC-enabled IoT environments. Because resources at the edge of the network are shared between application brokers [12, 13], the research problem is transformed to a lexicographical ordering bi-objective coalition structure generation (CSG) problem. Accordingly, the model of games (partition form games) with negative externalities [13] is adopted, and a novel deterministic anytime algorithm, called the Integer partition (IP) based minimum-cost adaptive policy control plan, is proposed to guarantee that, for specific planned time-periods, real-time EdgeIoT application instances will meet deadline. The novel contributions of this work to literature include:

- An SDN-based InterCloud architecture for emerging real-time MEC (edgeIoT) applications.
- An adaptive and exact Policy-based QoS control framework described using principles in coalition games with externalities or partition form games (PFGs) for capacity management in the SDN-based InterCloud platform.
- A minimum-cost dynamic and anytime control algorithm based on integer partitioning to iteratively explore policy subspaces and improve QoS concerned with IoT application latency.

The rest of this paper is organised as follows. Section II presents an overview of techniques related to resource management. Section III describes the SDN-based architecture for the InterCloud application scenario. Section IV provides a detailed description of the SDN-based adaptive capacity management framework. Section V outlines the design of experimental testbeds for emergent MEC paradigms. Section VI presents the results of experiments and discussion. Section VII concludes the paper.

II. OVERVIEW OF RESOURCE MANAGEMENT TECHNIQUES

Resource management is subject to incessant and sometimes unpredictable interactions with the environment. Managing resources in clouds typically requires complex policies and decisions for multi-objective optimisation. The quality of service (QoS) is that aspect of resource management that's probably the most difficult to address and, at the same time, possibly the most critical to the future of cloud computing. The resource management techniques for interactive workloads (Web services, for example) involve flow control and dynamic application placement, whereas those for non-interactive workloads are focused on scheduling [17].

A. Maintaining the Integrity of the Specifications

Grid ARS [18], an advanced reservation-based resource management framework, was developed using common web services technologies and standards and provides four services that address resource management, resource allocation planning, provisioning and monitoring of the constructed virtual infrastructure.

Stavrinides & Karatza [19] proposed a list scheduling heuristic for the scheduling of real-time workflow applications in a heterogeneous PaaS (or SaaS) cloud that incorporates imprecise computations and bin packing techniques. The scheduling approach has two objectives: (a) to guarantee that all applications will meet their deadline, providing high quality results and (b) to minimise the execution time of each workflow application and thus the cost charged to the user. The approach is compared to a baseline list scheduling algorithm via simulation, for workflow applications with various (communication to computation) ratios.

SDN-based resource allocation schemes [20] were proposed in ultra-dense orthogonal frequency division multiple access (OFDMA) multicell networks. The proposed scheme uses central resource management architecture to obtain the global information and finds out a sub-optimal solution with proportional fairness for resource allocation. An SDN-enhanced Job Management System [21] was developed as a network-aware resource management system (RMS) that offers a framework whereby administrators prescribe their own resource provisioning strategy using the information on both computing and network resources in a cluster system [22]. Furthermore, reference [22] explored an adaptive network resource allocation method with which network resources can be reallocated to jobs in execution, in response to the change of process placement occurred at job dispatch and termination events. In reference [23], a minimum-control-latency optimised algorithm based on greedy controlling pattern design was developed. The ideas and mechanisms are illustrated using the Internet2 OS3E topology compared to average-latency-optimized placement and worst-case-optimised placement. Results suggest the minimum-control-latency-optimised approach can improve the imbalance when partitioning SDN domains and achieve the maximum number of nodes per controller controlled. Reference [24] suggested that SDN can be coupled with Network Function Virtualisation (NFV) and cloud computing. Their effort proposed a simple and general SDN-IoT architecture with NFV implementation along with specific choices on where and how to adopt SDN and NFV approaches to address the new challenges of the IoT.

A feedback loop-based control scheme implemented as an SDN application named CPMan, was proposed in [25]. To adaptively tune the flow instantiation period to lower down the SDN control plane overhead, while always maintaining the data plane memory utilisation around a certain threshold. The control scheme was deployed on top of an SDN controller.

SLAM [26] is a latency monitoring framework that dynamically sends specific probe packets to trigger control messages from the first and last switches of a path to a centralised SDN controller. SLAM then estimates the latency distribution along a path based on the arrival timestamps of the control messages at the controller.

Bernstein & Vij [27] proposed the working of an InterCloud system, in terms of the details of an Intercloud Federation API, which transits the signaling network [27]. The federation API was used to dynamically provision a Software Defined Network (SDN) based Virtual Private Cloud (VPC) using Virtual Private networks (VPN), creating the federating bearer network for the transparent federation. The Intercloud Federation API is based on a semantic definition of resources, Service Level Agreements (SLA), and Bearer Network Provisioning Metadata [27, 28].

Cloud brokering is an important feature of Intercloud computing, which plays its role in terms of resource management, service discovery, service-level agreement negotiation and match-making between service providers and customer(s). A service-oriented dynamic resource management model [29], which covers cloud service consumer characteristics, was implemented and validated using the *CloudSim 3.0.3* toolkit. The method was also evaluated on Google cluster trace comprising 12,000 machines. Reference [30] developed a distributed control algorithm-based approach for performance management of services hosted in distributed cloud broker environments. Their research effort introduced a novel negotiation approach between the broker and the various cloud providers for optimised allocation of resources through interactive bidding in cloud computing environment. A generic architecture [31] was proposed for a Cloud service broker operating in an Intercloud environment by using Cloud standards. The goal of the broker is to find the most suitable Cloud provider while satisfying the users' service requirements in terms of functional and non-functional Service Level Agreement parameters. They focused especially on the incorporation of expected SLA management and resource interoperability functionalities in the broker. The proposed architecture was validated and evaluated using a realistic simulation testbed.

B. *Capacity Management and Load Balancing*

Capacity management involves adjusting the resources to meet demand or load comprising individual instances. Here, an instance refers to a service activation. ElastiCon [32], an elastic distributed SDN controller architecture, periodically monitors the load on each controller, detects imbalances, and automatically balances the load across controllers by migrating some switches from the overloaded controller to a lightly-loaded one. This way, ElastiCon ensures predictable performance even under highly dynamic workloads.

A flow migration approach to dynamically managed link and switch resources in an SDN-based virtualised environment was presented in [33]. The technique extended a floodlight controller by adding an application module which monitors the resource costs of mapped virtual links, as well as average load of the substrate links and switches.

A capacity allocation algorithm [34] was proposed to ensure SLA and handle fluctuating workloads. The allocation algorithms interact with geographically dispersed resource controllers and can redirect the load whenever congestion is present in the network. Moreover, it requires that an application is run on multiple VMs and workload is evenly distributed on the VMs. As the workloads vary, a workload predictor is used to forecast future workload requirements and resource capacity is changed based on resultant load forecasts. Amokrane et al. [35] proposed an architecture that integrates GPON management with an SDN controller. The framework introduces programmability and dynamic adaptation for grade PON (GPON) networks in response to traffic shifts in the network. Furthermore, an ILP formulation for the problem of deployment and dynamic traffic steering and capacity allocation was presented. To alleviate the time complexity of solving the ILP in large networks, they proposed a heuristic algorithm that achieves traffic steering on existing GPON deployments.

C. *Autonomous QoS Resource Management Policies*

In Cloud resource management, autonomic policies are of great interest due to the scale of the system, the large number of service requests, the large user population and the unpredictability of demand. The ratio of the mean to the peak resource needs can be significant [17].

Adami et al. [36] designed and developed a new network control application for QoS provisioning on top of the Floodlight controller. Their research modified the routing algorithm in order that link cost changes according to traffic load and more refined functions to handle traffic could be introduced. Bari et al. [37] presented the design and implementation of PolicyCop, an autonomic QoS policy enforcement framework based on SDN. Necessarily, PolicyCop provides an interface for specifying QoS policies and exploits the northbound API of SDN controller to enforce them. PolicyCop takes advantage of control applications to monitor the compliance of the policies and autonomically adapts the control plane rules with changing traffic conditions.

There is a substantial body of work in agent-based cloud computing that provides empirical evidence to show that multi-agent systems are appropriate software tools for automating complex interactions within an InterCloud [6]. Economic encounters between clouds in InterCloud were modelled as a coalition game [38]. To enable Clouds, discover and select their coalition members and to fairly divide the payoff of the InterCloud coalition, the effort devised a novel four-stage cloud-to-cloud interaction protocol that governs how cloud agents join coalitions, and a strategic profile of the cloud agents that converges to a sub-game perfect equilibrium.

To the best of the authors' knowledge, no new architecture has emerged to ensure low latency QoS related to capacity sufficiency in SDN-based InterCloud environments for real-time IoT (or big data) applications. This effort, therefore, models resource capacity provisioning between Clouds in SDN-based InterCloud platforms for future MEC or EdgeIoT applications as coalition games with externalities or partition form games (PFGs). The method of investigation in this research proposes a perspective that differs from the approach in [38]. Firstly, in our proposed approach, the coalition formation process, or the solution concept which involves Clouds forming an InterCloud coalition is done exogenously (automatically by the system administrator) rather than self-interested clouds negotiating and establishing agreements with other clouds (i.e. endogenously) to meet its own objectives and to optimise its own payoff. Secondly, this investigation also accounts for externalities [12, 39, 40] that may exist as a` result of disjoint resource coalitions.

III. SDN-BASED INTERCLOUD ARCHITECTURE

In our proposed architectural design, we focus on adaptive (autonomous) resource capacity management in SDN-based

InterCloud environments to control the operational latency of real-time IoT applications at the control plane. Fig 1 depicts the proposed architecture of the SDN-based Intercloud application scenario for future MEC environments. As with a typical SDN architecture, the platform has in addition to management & administration, three main planes namely: application, control, and data. The application plane can be viewed simply as comprising client-centric SDN-based IoT application instances or demand, in this paper, we omit discussing it in further detail.

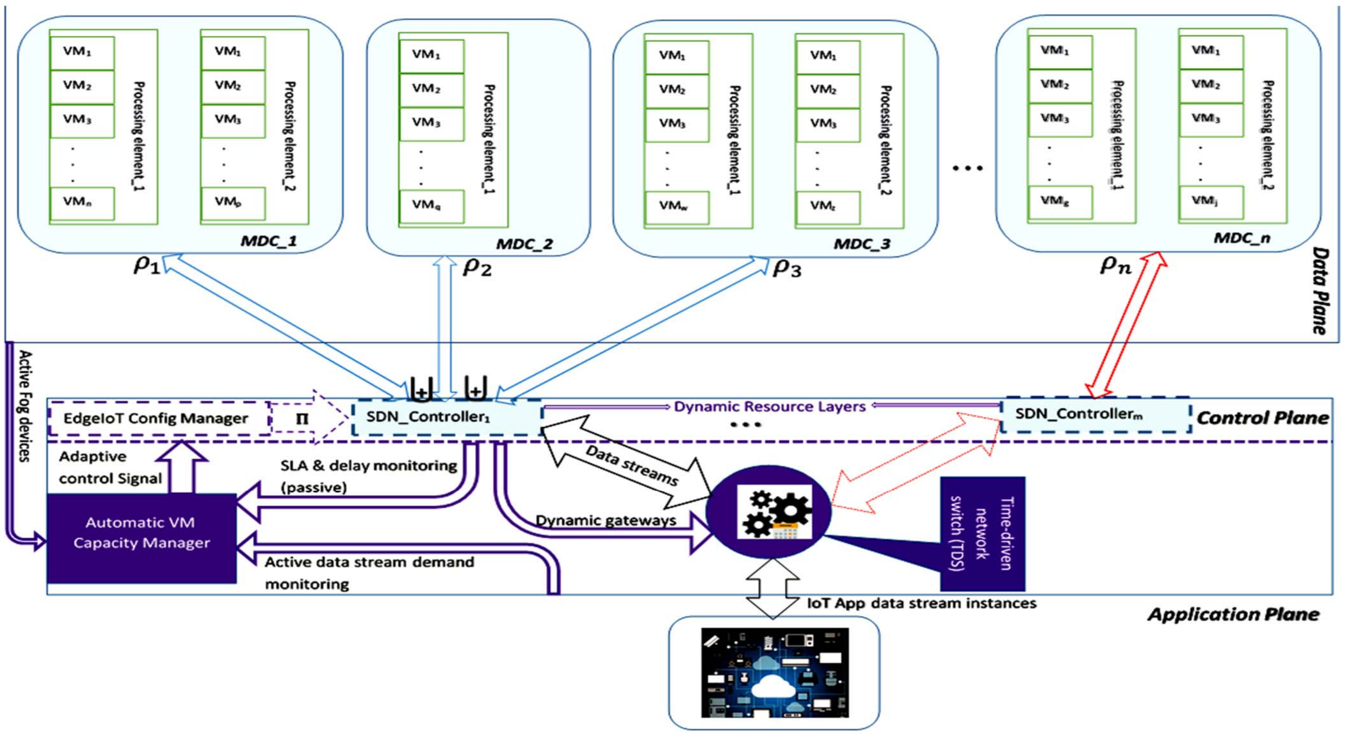


Fig.1. The SDN-based InterCloud Architecture

A. Control Plane

In the SDN control model, controllers, also called orchestration platforms act as a network packet "brokers" between client IoT applications and network processing elements. Application brokering is implemented either in a centralised entity or by the cloud providers in the case of InterClouds [41]. In this effort, we define dynamic SDN controllers or SDN brokers logically as centralised entities generally used to support loosely coupled IoT applications in which independent components work together to accomplish a real-time task. The InterCloud-enabled application controllers in this research are dynamic and centrally-located in the control plane of the SDN-based InterCloud environment between IoT application plane and fog data plane. We discuss briefly two brokering mechanisms as it concerns our implementation of SDN-based InterCloud application controllers.

- SLA-based packet brokering mechanisms: Service level agreement (SLA) is signed between service providers and clients with the intention to provide the best suitable performance for consumers. Transparency is the focus of the SLA-based brokering approaches, that is, resource provisioning is done automatically. Also, clients have no control over how their applications are provisioned across fog infrastructure. Therefore, there must be a certain level of trust between the client and providers. The requirement for application provisioning is normally achieved using formal SLA specifications and services.

- Directly-managed packet brokering mechanisms: Directly-managed brokering mechanisms are mostly used in situations when there is no mediation between client IoT applications and the set of fog data centers. In this case, it is the responsibility of directly-managed brokers (hosted separately) to monitor the performance of client IoT applications. Here, the application brokers should be developed in such a way that availability and reliability requirements are met.

B. Data Plane

The provider environment or SDN data plane consists primarily of fog micro data centers (MDCs) as network processing elements (for fast processing of data streams) provided by intermediary (proximate) cloud infrastructures. More specifically, let's assume we can increase resource capacity by federating (Intercloud application brokering) virtual resources from n small-scale fog data centers or cloudlets within close geographic proximity. Application traffic processing nodes in the InterCloud SDN environment is defined as the set $\mathrm{E} = \{\rho_1, \rho_2, \ldots, \rho_n\}$. This set denotes singleton near proximity processing nodes, where ρ_i represents the provisioned capacity (of network processing elements) of resources or

Identify applicable sponsor/s here. If no sponsors, delete this text box (*sponsors).*

application-specific virtual machines (VMs) in fog micro data center (MDC) i $(i \leq n)$.

C. Management and Administration

The proposed intelligent management layer is made up of the adaptive VM capacity manager, the policy configuration manager, and SDN monitoring interfaces. The automatic VM capacity manager forms the core of the management layer whose fundamental objective at any given time is to make optimal decision about the best or most appropriate policy to execute. The policy configuration manager is responsible for dynamic behavior in SDN controllers driven by decision control signals from the VM capacity manager.

Monitoring in the SDN-based environment should be continuous to facilitate decisions as part of overall resource utilisation optimisation. Importantly, monitoring may be carried out passively or actively. In passive monitoring, there are one or more entities collecting information. The entity may continuously send polling messages to nodes asking for information or do this on demand when necessary [42]. On the other hand, monitoring is active when nodes are autonomous and may decide when to send asynchronously state information to some central entity. Both active and passive monitoring is used simultaneously to improve the policy. In this case, it is necessary to synchronise updates in repositories to maintain consistency and validity of state information.

The architecture also proposes a *traffic router,* capable of forwarding big data packets in real-time, that is centrally-placed between the application and control planes. The router is designed to support fast processing of IoT application streams. The concept of *IoT traffic gateways* is the fundamental abstraction provided by the IoT router situated at the edge of the network. Conceptually, SDN controllers' function to forward IoT big data streams to one or more processing element(s) in E through *dedicated gateways* specifically set up to mirror the *economics* of shared resources at the edge of the network. The IoT routing policy can be implemented to support either single node or multiple nodes. This paper only focuses on the design of an adaptive capacity manager based on an implementation of the single node IoT router.

IV. The Deterministic Approach for QoS-Aware Adaptive Capacity Management

Formal models are mostly used to describe the behavior of systems (system dynamics) and control them via processes for computation— sequential and parallel, deterministic and non-deterministic—for instance, controlled-markov chains (CMC) and Petri nets. In execution, Petri-nets are largely non-deterministic models and more suited to managing resources for interactive workloads like flow control and dynamic application placements. However, in this research, we present a deterministic anytime scheduling approach to model the resource capacity management problem for non-interactive workloads in SDN-based InterCloud environments.

We propose that the game-theoretic (coalition games) system dynamics model for the QoS-aware autonomic capacity management framework in the control plane of the SDN-based InterCloud architecture is a 5-tuple Markov Decision Process $(\wp, \mathbb{T}, \boldsymbol{Q}, \boldsymbol{\eta}, \boldsymbol{q_0})$.

Briefly, $\wp$ represents the decision space of capacity scheduling policies in the SDN-based InterCloud environment, $\mathbb{T}$ is time; Q is a set of environment states each combining resource capacity, demand and capacity policy. q_0 is the initial state; and η is a transition function mapping to the next state of resource functions, policies.

$\wp = (\boldsymbol{G}(\boldsymbol{V}, \boldsymbol{\xi}), \mathbb{C})$, the network's capacity scheduling or management policy space is represented as a graph. Particularly, we say, G is the Integer partition (IP) based policy management graph, and V is the set of vertices or policy subspaces of G, each comprising at least one policy. The set ξ of edges in G represent coalition mergers (InterClouds) particularly to increase resource capacity, and splits (to reduce capacity). $\mathbb{C}$ is the search cost metric that simply computes the cost of searching a policy subspace V in G, equal to the aggregate number of capacity policies in the policy subspace V.

The Time $\mathbb{T} = (\boldsymbol{T}, \boldsymbol{\Sigma}, \boldsymbol{\Upsilon}, <)$ is a dimensionless measured space with strict total ordering [43]. T is a set of time periods with at least one element κ_0, that is, $\kappa_0 \in T$. T can be viewed as a set of abstract time steps $T = \{\kappa_0, \kappa_1, \kappa_2, \ldots\}$.

Σ is defined as σ-Algebra over time-period T or more simply as a non-empty set of subsets of T closed under union and complement with respect to T.

Υ is the measure of peak data stream traffic of the real-time IoT application in T. For example, let $t_1 \in \Sigma$ then $\Upsilon(t_1) = \mu$. We define μ as the peak number of application data stream instances. The 'less than' $(<)$ operator represents a strict total order on T.

The set of environment states $\boldsymbol{Q} = \{\boldsymbol{q_0}, \boldsymbol{q_1}, \boldsymbol{q_2}, \ldots\}$ where q_0 is the initial state and each $q_j = (R_{qj}, \Pi_{qj})$. Furthermore, R_{qj} is a resource function that comprises resource capacity (the set of compute nodes E) and peak application stream demand whereas Π_{qj} represents the capacity management policy.

The transition function $\boldsymbol{\eta}: \boldsymbol{T} \rightarrow \boldsymbol{Q}$ maps a specific time to specific environment states. In other words, given any discrete time, the function translates it to a state of the environment.

A. Problem Statement

In a network virtualisation environment (NVE), virtual networks (VNs), composed of virtual routers and virtual links, are deployed on a shared physical network, called substrate network (SN). The main NV problem consists of choosing how to allocate a VN over an SN, meeting requirements and minimising resource usage of the SN. Finding the capacity policy that maximises QoS related to latency requirements in the SDN-based InterCloud application environment is the coalition structure generation (CSG) problem. Generally, this may involve searching the space of all possible capacity policies denoted as Π^E to determine the policy that maximises for peak demand in time $t \in \Sigma$, the overall QoS related to latency in real-time IoT applications.

B. Resource Capacity Links and Capacity Policies

Conceptually speaking, a resource capacity link is a non-empty subset of E. Specifically, each SDN application controller initiated in the control plane of the environment maintains a stored resource capacity link, in say, a linked list data structure, to resources in one or more nodes of E.

Accordingly, any instance of an SDN controller's resource capacity link L is a member of the power set of E, **i.e.** $\mathrm{L} \in \mathrm{P(E)}$. Since, the set of processing function nodes is *finite,* $|\mathrm{E}| = \mathrm{n}$, therefore, $|\mathrm{P(E)}| = 2^{\mathrm{n}}$. Furthermore, using set notation, we can describe the set of all possible controller resource links denoted $\hat{L}$ as,

$$\hat{L} = \{L: L \subseteq E, L \neq \emptyset\}. \quad (1)$$

For simplicity, we can rewrite $\hat{L}$ as $\hat{\mathrm{L}} = \mathrm{P(E)} - \{\emptyset\}$. More specifically,

$$|\hat{\mathrm{L}}| = |\mathrm{P(E)}| - |\{\emptyset\}| = 2^{\mathrm{n}} - 1 \quad (2)$$

Let $\hat{L}_r$ denote all the possible resource links made up of r network processing elements. It also follows that the number of possible resource links in $\hat{L}_r$ can be described as the binomial coefficient (or combination) of n and r, computed as,

$$|\hat{L}_r| = \binom{n}{r} \quad (3)$$

Also, assume $\hat{L}_{2\le r\le n}$ represents all possible resource capacity links of SDN-based InterCloud application controllers. It can be deduced that $|\hat{L}_{2\le r\le n}| = \sum_{r=2}^{n} |\hat{L}_r|$, and since $|\hat{L}| = \sum_{r=1}^{n} |\hat{L}_r|$, this implies using eq. (3) and (2) that, $|\hat{L}_{2\le r\le n}| = |\hat{L}| - |\hat{L}_1| = |\hat{L}| - \binom{n}{1} = |\hat{L}| - n$, therefore,

$$|\hat{L}_{2\le r\le n}| = 2^n - n - 1 \quad (4)$$

Given a state $q \in Q$ of the network environment, any *exhaustive* partition of E is known as a coalition structure. This partition or coalition structure comprises controllers with mutually isolated resource capacity links in the SDN-based InterCloud environment and is referred to as the capacity policy or more simply, a policy denoted as Π.

$$\Pi = \{L_1, L_2, \ldots, L_{|\Pi|}\},\ 1 \le |\Pi| \le n \quad (5)$$

In addition to the non-empty subset constraint for constituting resource capacity links, a capacity policy Π in the SDN environment state satisfies the following simple rules:

(i) $\bigcup_{j=1}^{|\Pi|} L_j = E,\ \forall L_j \in \Pi$

(ii) $L_p \cap L_q = \emptyset,\ p, q \in \{1,2,\ldots,|\Pi|\},\ p \neq q,\ \forall L_p, L_q \in \Pi$.

Let Π^E denote the set of all capacity policies and the set of policies containing exactly m resource capacity links is designated as Π_m^E. Thus, the number of possible policies is computed as the n^{th} bell number,

$$|\Pi^E| = \sum_{m=1}^{n} |\Pi_m^E| \quad (6)$$

Where $|\Pi_m^E|$ is evaluated as,

$$|\Pi_m^E| = (1/m!) \sum_{k=0}^{m-1} (-1)^k \binom{m}{k} (m-k)^n \quad (7)$$

TABLE I. EXPONENTIAL GROWTH IN CAPACITY LISTS AND POLICIES WITH AN INCREASE IN NETWORK PROCESSING ELEMENTS

n	1	3	5	7
$\lvert\hat{\mathrm{L}}\rvert$	1	7	31	127
$\lvert\Pi^E\rvert$	1	5	52	877

C. The Integer Partition (IP) Policy Management Graph

The space representation of policy set Π^E that involves grouping policies into levels based on the number of controller resource links it contains is called the coalition structure (CSG) graph [12]. Specifically, the CSG graph is an undirected graph, in which every node would represent a capacity policy Π, and policies are categorised into levels based on the number of resource links in it. For instance, a capacity management policy comprising distinct 4 controller resource links should be found on level 4 of the CSG graph. Also, an edge connects two capacity policies in the CSG graph if and only if, the two policies are situated in levels that are consecutive, and policies in a higher level can be attained from that in the lower level by merging two resource capacity links into one. However, because policies of different capacity can be located on the same level, search becomes unstructured. Searching policies level-wise with a guarantee for increased or decreased resource capacity becomes impossible.

To address this, the approach to capacity scheduling in this paper adopts a novel representation of the policy space Π^E called the Integer partition (IP) graph [13, 44]. We propose that the set of all possible management policies, Π^E in the environment is grouped into disjoint policy subspaces, each of which is represented by an integer partition of n, number of active fog datacenters in E. For clarity, an Integer partition of n consists of *integer parts*, the sum of which equals n.

In the IP policy graph representation, every vertex in the graph represents a policy subspace that comprises policies, nodes are categorised into n levels (partition spaces) denoted $I_1, I_2, \ldots I_n$ Partition space $I_k (k \le n)$ contains the policies comprising k broker (or controller) resource links [13, 45]. Specifically, if G^n represent the IP graph of the SDN-based environment comprising n fog network processing nodes then the vertex or policy subspace $\Pi_{[|L_1|,|L_2|\ldots,|L_q|]} \in I_q$, the partition subspace that consists of policies with q distinct or non-intersecting controller resource links such that, $|\mathrm{L}_1| + |\mathrm{L}_2| \ldots + |\mathrm{L}_q| = |\mathrm{E}| = \mathrm{n}$, and $q \le n$. For example, Fig.2 shows the IP graph G^4 consisting of vertices, policy subspaces for 4 fog data centers in the SDN data plane is enumerated as,

$$G^4 = \{\Pi_{[4]}, \Pi_{[2,2]}, \Pi_{[1,3]}, \Pi_{[1,1,2]}, \Pi_{[1,1,1,1]}\} \quad (8)$$

D. Monitoring the SDN environment state

In the SDN-based environment, we propose active monitoring of IoT application stream demand to guarantee reliability about QoS. However, SLA and latency delay monitoring is done passively.

1) Delay monitoring of Policy based on controller complexity

We propose a straightforward design (algorithm 1) of the generic cyclic behavior for SDN controllers to forward IoT data streams. The design complexity, as proposed in *algorithm 1*, of the generic stream processing behavior for application controllers in the SDN environment can be easily measured in terms of the number of forwarding nodes in the packet broker's (SDN controller) resource capacity link L. However, it would

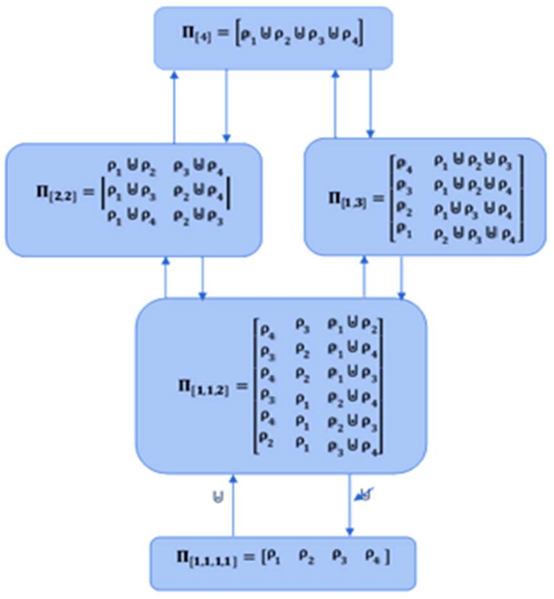

$\Pi_{[4]} = [\rho_1 \uplus \rho_2 \uplus \rho_3 \uplus \rho_4]$
$\Pi_{[2,2]} = \begin{vmatrix} \rho_1 \uplus \rho_2 & \rho_3 \uplus \rho_4 \\ \rho_1 \uplus \rho_3 & \rho_2 \uplus \rho_4 \\ \rho_1 \uplus \rho_4 & \rho_2 \uplus \rho_3 \end{vmatrix}$
$\Pi_{[1,3]} = \begin{bmatrix} \rho_4 & \rho_1 \uplus \rho_2 \uplus \rho_3 \\ \rho_3 & \rho_1 \uplus \rho_2 \uplus \rho_4 \\ \rho_2 & \rho_1 \uplus \rho_3 \uplus \rho_4 \\ \rho_1 & \rho_2 \uplus \rho_3 \uplus \rho_4 \end{bmatrix}$
$\Pi_{[1,1,2]} = \begin{bmatrix} \rho_4 & \rho_3 & \rho_1 \uplus \rho_2 \\ \rho_3 & \rho_2 & \rho_1 \uplus \rho_4 \\ \rho_4 & \rho_2 & \rho_1 \uplus \rho_3 \\ \rho_3 & \rho_1 & \rho_2 \uplus \rho_4 \\ \rho_4 & \rho_1 & \rho_2 \uplus \rho_3 \\ \rho_2 & \rho_1 & \rho_3 \uplus \rho_4 \end{bmatrix}$
$\uplus$
$\Pi_{[1,1,1,1]} = [\rho_1 \quad \rho_2 \quad \rho_3 \quad \rho_4]$

Fig.2. IP-based Policy Management Graph for SDN environments comprising 4 network elements.

be more appropriate (comprehensible) if the complexity can be quantified in terms of amortised cost or time. To do so, we propose, based on an agent-based design paradigm, that the cost attributed to latency delay in application controllers' is tightly coupled to its behavioral complexity. As such, we assume, theoretically at least, that this cost is linearly *proportional* to the size of the controller's resource link $|L|$ or number of data forwarding paths in L. Besides, the worst-case behavioral complexity of a capacity policy Π can be estimated as a linear constant real-time function φ_t $(t \in \Sigma)$ calculated as,

$$\varphi_t(\Pi) = arg \max_{L \in \Pi}(|L|) \tag{9}$$

Furthermore, assume ϑ is the worst-case stream forwarding start time of controllers using resource links of maximum size in policy Π of the SDN environment, then, $\vartheta \propto \varphi_t(\Pi)$. Suppose β_0 is the forwarding start time of controllers using resource links L such that $|L| = 1$, thus,

$$\vartheta = \beta_0 . \varphi_t(\Pi) \tag{10}$$

2) SLA monitoring of Policy

Here, we describe performance monitoring related to low latency requirements in SLA contracts for real-time big data applications. The value or capacity sufficiency γ_t for any policy Π is measured by monitoring the rate of success for controllers meeting deadline constraints for real-time application instances. Let embedded functions $\omega^L{}_{t,\Pi}$ and $\tau^L{}_{t,\Pi}$ designate the number of deadline successes and violations achieved by any controller using resource link L in policy Π $(L \in \Pi)$ for processing the peak application stream demand in measured time $t \in \Sigma$. Because a controller's resource link L may co-exist with other controller links in the environment state, let $\bar{L}$ denote the set of the resource links used by other controllers in policy Π, i.e. $\bar{L} = \Pi - L$. For $t \in \Sigma$, we denote $\overleftarrow{e}^{\omega}_{t,\bar{L}}$ and $\overleftarrow{e}^{\tau}_{t,\bar{L}}$ as the negative externality [12, 38] (successes and violations) acting upon L

Algorithm 1: Generic packet brokering behavior for SDN Application controllers

```
Input:  Resource capacity Link L (Linked List)
1.    get all traffic processing nodes $\rho \in L$
2.        For all processing nodes $\rho$
3.            Forward incoming IoT streams for processing.
```

Fig. 3. SDN-based InterCloud Application Controller behavior

by $\bar{L}$. Also, in the absence of externality for a controller resource link L in policy Π, let $\omega^L{}_t$ and $\tau^L{}_t$ denote the number of SLA deadline successes and violations. Applying the terms previously defined, (11) and (12) show how to evaluate embedded functions $\omega^L{}_{t,\Pi}$ and $\tau^L{}_{t,\Pi}$ when externalities are present in time.

$$\omega^L{}_{t,\Pi} = \omega^L{}_t + \overleftarrow{e}^{\omega}_{t,\bar{L}} \tag{11}$$

$$\tau^L{}_{t,\Pi} = \tau^L{}_t + \overleftarrow{e}^{\tau}_{t,\bar{L}} \tag{12}$$

Also, for weak negative externality [13] over time t, we set $\overleftarrow{e}^{\omega}_{t,\bar{L}}, \overleftarrow{e}^{\tau}_{t,\bar{L}} = 0$. Consequently, we evaluate the capacity sufficiency γ_t for any controller using resource link L as,

$$\gamma_t(L) = \omega^L{}_{t,\Pi} / (\omega^L{}_{t,\Pi} + \tau^L{}_{t,\Pi}) \tag{13}$$

And, the capacity sufficiency γ_t for any policy Π is computed as,

$$\gamma_t(\Pi) = \textstyle\sum_{L \in \Pi} \omega^L{}_{t,\Pi} / (\sum_{L \in \Pi} \omega^L{}_{t,\Pi} + \tau^L{}_{t,\Pi}) \tag{14}$$

E. Policy Optimisation

A primary characteristic of multi-criteria optimisation (MCO) problems is the need for a decision maker's intervention in the optimisation process**.** The goal of the adaptive capacity manager proposed in this research is to maximise QoS objectives concerned with the low latency requirements of real-time IoT applications. This implies maximising capacity sufficiency (γ_t) to guarantee latency requirements in SLA contracts as well as reducing latency delay (φ_t) related to the design complexity of application controllers. Intuitively speaking, for time-shared scheduling of resources in hosts machines, the capacity sufficiency is most critical to the performance of real-time big data MEC applications. Interestingly, this is well under the control of the decision maker as compared to the latency delay (cost) in SDN controllers which is a function of the network packet broker's design complexity.

Therefore, we propose the a priori Multi-Criteria Optimisation (MCO) method called lexicographic method [46]**,** which assumes that objectives can be arranged in a hierarchy of preference or importance. The initial step in this method is to categorise the objective functions into various levels based on their importance. The highest level is the most important whilst the lowest level is the least important.

Accordingly, we describe the bi-objective maximisation optimisation problem for QoS control such that the capacity sufficiency γ_t is the highest priority or objective whereas the inverted delay $\varphi_t{}^{-1}$ is the least priority. Precisely, let f_1 and f_2 denote the objectives with the highest and least importance respectively,

$$f_1 = \gamma_t \tag{15}$$

$$f_2 = \varphi_t{}^{-1} \tag{16}$$

Although a solution that simultaneously optimises all objective functions is most desirable, except for trivial cases, such a solution is not usually feasible. The pareto solution based on the planner's implicit preferences towards individual planning criteria is sought through trade-offs. Due to its simplicity and effectiveness, the conventional approach for dealing with the MCO aspect of the problem is to use the scalarisation approach, in which multiple objective functions are combined to a single objective function. Thus, we apply a common scalarisation approach known as the weighted sum method,

$$\boldsymbol{\Pi}^* \leftarrow \boldsymbol{max}_{\Pi \in G^n}\left(\textstyle\sum_{i=1}^{2} \boldsymbol{\alpha_i \cdot f_i}\right) \tag{17}$$

$$\textbf{\textit{Subject to:}}\ \textstyle\sum_{i=1}^{2} \boldsymbol{\alpha_i} = \mathbf{1},\ \boldsymbol{\alpha_1 > \alpha_2 > 0}$$

F. The IP Anytime minimum-cost Policy control

From the SDN-based InterCloud architecture in Fig. 1, it is obvious that any policy control plan for the capacity manager is quite intrinsic to the packet forwarding strategy of the centralised IoT router. To simplify communication, we define a *static forwarding policy for the single node IoT router* as simple as directing communication of *IoT application traffic* through the dedicated gateway of the SDN controller whose resource link is of *maximum size in the active state policy*. It is worthwhile to note that for the single node implementation of the IoT router, only one SDN-based InterCloud application controller is active at any given time, and as such, in this case every externality in the system equals zero and the administrator puts the inactive fog devices on standby.

Furthermore, we can use the forwarding policy of the single node IoT router to prune unnecessary policy subspaces from the IP policy management graph G^n. We are only interested in policy subspaces that are guaranteed to reduce the ratio bound β of the solution from the optimal to 1. For games with negative externalities, the minimum set of policy subspaces required to establish a ratio bound β from the optimal (in this scenario, the ratio bound $\beta = 1$) is $\{\Pi_{[1,\dots,1]} \cup \dots \cup \Pi_{[n-1,1]} \cup \Pi_{[n]}\}$ [13, 38]. Therefore, the remaining subspaces are not worth exploring and should be pruned from G^n. The reason for this is because, in this set, every possible SDN application controller resource link of *maximum size* appears in a unique policy except for the resource layers of size 1, these all appear in a single resource capacity policy. Thus, the total number of policies$|\Pi^E|$ that need to be searched is reduced to $2^n - n$. The effort in [13, 38] identified the set of IP subspaces that needs to be searched (to establish a bound) but does not particularly specify how this search takes place. Considering that all IP-based policy subspaces must be explored to guarantee QoS, we propose, in this, paper, an uninformed search algorithm in which policy subspaces of cumulative minimum cost are explored, referred to as uniform cost search. At any point in the execution, the algorithm always expands a node (policy subspace) which has a cost not greater than the cost of any other path or node in the IP graph.

Algorithm 2 provides a listing of the proposed anytime IP minimum-cost policy control plan for the VM capacity manager. The procedure is based on our implementation of the single node IoT router. Anytime algorithms terminate before end of execution implies a workable solution candidate to the problem. The purpose of such algorithms is to improve the solution with time. In the worst case, the procedure applies the uniform cost search algorithm to improve the solution quality by searching policy subspaces of minimum-cost using a priority queue data structure P, the priority queue P encapsulates methods for enqueue and dequeue operations placing the search cost $\mathbb{C}$ of policy subspaces as the priority. The search is carried out for better solutions by exploring policies in the policy subspace of the IP graph. $Search^1$ denotes a single policy evaluation of the subspace is sufficient whereas $Search^*$ implies in the worst case, it is necessary to explore all policies in the subspace.

V. EXPERIMENTAL TESTBED

To fulfil the low latency requirements of IoT big data applications processed at the edge of the network they need to be checked for their performance i.e. turnaround time so that performance is within constraints specified in SLA contracts. CloudSim [14-16] is essentially a J*ava-based framework* that enables researchers and industry-based developers to focus on specific system design issues worth investigating while ignoring the low-level details related to data center infrastructures and services. The Cloudsim 4.0 toolkit provides a generalised, and extensible simulation framework that facilitates seamless modelling, simulation, and experimentation of emerging Cloud computing infrastructures and application services. This section provides in sufficient detail description of the components for the SDN-based EdgeIoT (MEC) testbed.

A. The CloudSim Experiment Design

The Cloudsim framework has been used in several other researches to model and simulate cloud services and infrastructure [16]. Its framework provides an adequate abstraction for describing workloads and data center infrastructure. In addition, CloudSim toolkit provides a host of Classes that enabled us model cloud federations which

Algorithm 2: IP Minimum-cost Self-Tuning control

```
Input: $n$ – $number\ of\ fog\ data\ centers,\ P - search\ cost\ priority\ Queue$
        $Flag$ – True, if VM capacity is same for all n fog nodes (false, otherwise)
        $G^n = \{\Pi_{[1\dots1]}, \dots, \Pi_{[n-1,1]}, \Pi_{[n]}\}$ – $Pruned\ IP\ policy\ graph$
1:  If (Flag)
2:      For $i = 1$ to $n$
3:          If ($i < n$)
4:            $Search^1\ \{\Pi_{[i,\dots,1]}\}$
5:          Else
6:            $Search^*\ \{\Pi_{[i]}\}$
7:          End If
8:      End For
9:  Else
10:    $Search^*\ \{\Pi_{[1,\dots,1]}, \Pi_{[n]}, \Pi_{[n-1,1]}\}$
11:    For $i = 2$ to $n - 2$
12:        $\mathbb{C}(i) \leftarrow \langle{n \atop i}\rangle$
13:        $P.add\_with\_priority(i, \mathbb{C}(i))$
14:    End For
15:     While $P$ is not empty
16:        $j \leftarrow P.Min\_cost()$
17:        $Search^*\ \{\Pi_{[j,\dots,1]}\}$
18:     End While
19: End If
```

Fig.4. Anytime IP Uniform-cost Policy control plan for VM capacity Manager.

constitute edge resource layers responsible for processing IoT applications. Particularly, the *DatacenterBroker* super *Class* implemented in Cloudsim supports application-specific brokering for implementing EdgeIoT resource layers (or SDN application controllers). However, the implemented class does not prevent overlapping SDN controllers at the edge of the network and hence does not suffice for implementing SDN controllers in this research. As such, to implement SDN controllers, it was necessary we extend the *DataCenterBroker Class* in Cloudsim to implement the *ResourceLayer Class (Fig.5) for the EdgeIoT (MEC) testbed.*

B. The Experimental Setup

This research designed and implemented some experimental test environments applying the agent based modelling and simulation framework for capacity management in Fig. 6, and discusses the results to evaluate or benchmark our proposed QoS-aware capacity management approach. The *Cloudlet* class in Cloudsim was used to represent an instance of the IoT big data application. The parameters for the agent-based model of the IoT application include task length, file size, output size, processing elements and utilisation model.

VM architecture for IoT application data stream instance is configured with parameters such as memory(Ram), bandwidth, millions of instructions per second (mips), Image size. VM-based fog micro data centers (MDCs) [47] are the network processing elements in the data plane of the SDN-based testbed and offers the various infrastructure resources to the

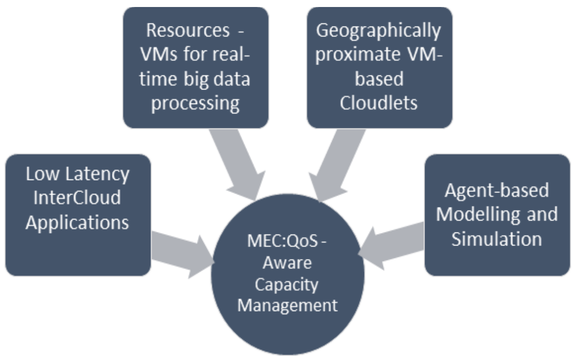
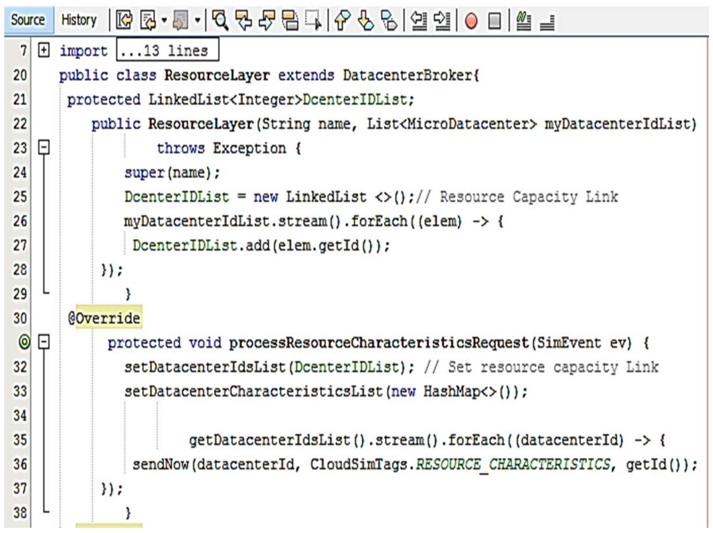

Resources - VMs for real-time big data processing
Geographically proximate VM-based Cloudlets
Low Latency InterCloud Applications
MEC:QoS - Aware Capacity Management
Agent-based Modelling and Simulation

Fig.6. Input framework for QoS-aware capacity management in MEC environments

applications. Fog MDCs were implemented by extending the DataCenter Class of CloudSim 4.0, using for example, compute and storage parameters. The resources offered may have different configuration, for instance, processor architecture, Operating system and Virtual machine manager. The parameters set for the agent based simulation of fog computing nodes are: number of fog cloudlets n, architecture, Operating system type, VM manager, RAM, processing capacity, time zone and processing cost/sec. Table 2 provides a summary of the settings for the experimental EdgeIoT testbed. In the design of preliminary experiment testbeds, we assumed homogenous infrastructure characteristics such as processing capacity (mips), pricing, storage and memory for the 4-fog data centers.



```
import ...13 lines
public class ResourceLayer extends DatacenterBroker{
 protected LinkedList<Integer>DcenterIDList;
    public ResourceLayer(String name, List<MicroDatacenter> myDatacenterIdList)
            throws Exception {
        super(name);
        DcenterIDList = new LinkedList <>();// Resource Capacity Link
        myDatacenterIdList.stream().forEach((elem) -> {
         DcenterIDList.add(elem.getId());
    });
        }
 @Override
     protected void processResourceCharacteristicsRequest(SimEvent ev) {
        setDatacenterIdsList(DcenterIDList); // Set resource capacity Link
        setDatacenterCharacteristicsList(new HashMap<>());

                getDatacenterIdsList().stream().forEach((datacenterId) -> {
         sendNow(datacenterId, CloudSimTags.RESOURCE_CHARACTERISTICS, getId());
    });
        }
```

Fig.5.Code snippet of the Resource Layer (Application Controller) implementation in Cloudsim 4.0 (NetBeans 8.0)

TABLE II. CLOUDSIM EXPERIMENT FOR EDGEIOT TESTBED

Simulation parameters	Values
VM specification	[4096MB, 1000, 3000, 1000MB]
EdgeIoT application	[75000, 30000, 30000, 2, Full]
Fog DCs	***[4, X86, Linux, Xen, 8192*2 MB, 2, 3000000 mips, +1:00GMT, 3.0]***
$\boldsymbol{\Upsilon(t \in \Sigma)}$	[2, 6, 10,14]
Service level latency requirement	less than 0.18 secs

VI. RESULTS AND DISCUSSION

In the experimental testbed designed to validate our approach, we set the processing start time ϑ to at least one-tenth smaller than the latency constraint in SLAs for the real-time EdgeIoT application. The main objective of experiments performed is to investigate the impact of dynamic SDN-based InterCloud application controllers (called Resource layers) on (i) the SDN controller forwarding start time (ii) the average EdgeIoT application latency, and (iii) the average application processing cost. The experiments were carried-out by varying the size of a centralised SDN controller resource link L for peak edgeIoT application demands $\Upsilon(t_0), \Upsilon(t_1), \Upsilon(t_2), \Upsilon(t_3)$ in planned time-periods $t_{i=0,..3} \in \Sigma$. Empirical results validated the policy optimisation process (Table III) for the adaptive VM capacity manager using the weighted-sum *scalarisation method*. The adaptive VM capacity manager provides QoS guarantees by determining for each planned peak IoT demand $\Upsilon(t)$ a control decision signal Π^*, the optimal solution to the lexicographical ordering bi-criteria maximisation problem using $\alpha_1 = 0.99$ and $\alpha_2 = 0.01$.

TABLE III. ANYTIME & DYNAMIC PROGRAMMING POLICY OPTIMISATION PROCESS

		IP CONTROL PLAN								CAPACITY MANAGER: BI-CRITERIA DECISION PROBLEM				CONTROL SIGNAL	
TIME-VARYING DEMAND	PEAK IOT APP INSTANCES $\Upsilon(t)$	SLA MONITOR f_1 $(\alpha_1 = 0.99)$				DELAY MONITOR f_2 $(\alpha_2 = 0.01)$				$\max_{\Pi}(\alpha_1 f_1 + \alpha_2 f_2)$				Π^*	
		$\lvert L\rvert = 1$	$\lvert L\rvert = 2$	$\lvert L\rvert = 3$	$\lvert L\rvert = 4$	$\varphi_t = 1$	$\varphi_t = 2$	$\varphi_t = 3$	$\varphi_t = 4$	$\Pi = 1$	$\Pi = 2$	$\Pi = 3$	$\Pi = 4$		
	(X2)	1	1	-	-	1	0.5	-	-	1	0.995	-	-	$\Pi = 1$	VM POLICY
	(X6)	0.33	1	1	-	1	0.5	0.33	-	0.337	0.995	0.9933	-	$\Pi = 2$	
	(X10)	0	0.6	1	1	1	0.5	0.33	0.25	0.01	0.599	0.9933	0.9925	$\Pi = 3$	
	(X14)	0	0.143	0.714	1	1	0.5	0.33	0.25	0.01	0.147	0.710	0.9925	$\Pi = 4$	

A. *SDN Application Controller Delay*

The experimental results provide evidence to suggest that the processing start time ϑ for SDN application controllers increases with the size of its resource link L *Fig. 7* shows a linear proportionality between the resource layer size or SDN controller resource link size and the delay in the processing start time ϑ. From equation (9), we can deduce that application

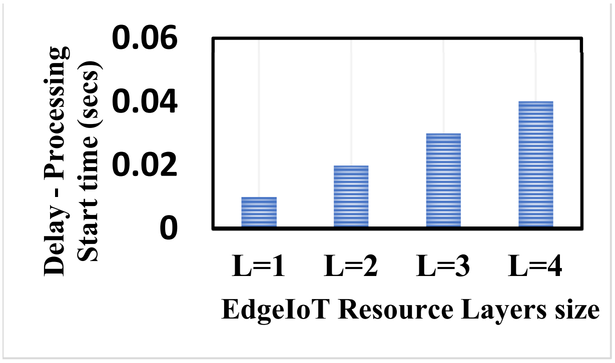


Fig.7. Forwarding delay for SDN-based InterCloud application controllers

brokers in VM resource layers $L = 1$ have the average processing start time $\beta_0 = \vartheta/|L| = 0.01$ secs. This also means we can estimate the forwarding start time or delay for any size of resource layer say $|L| = n$ as $\vartheta = 0.01 * n$ secs. In all the experimental testbeds varying the peak IoT application demand $\Upsilon(t)$ had no effect on the processing start time ϑ.

B. *EdgeIoT Application latency*

The results for average latency shown in *Fig.8* of low peak homogenous IoT application demand or instances worsens with an increase in the size of resource layers. For instance, the average latency for peak demand $\Upsilon(t_1)$ in resource layers $L = 2$ is marginally reduced compared to the average latency in resource layers $L = 3$ and $L = 4$. This marginal deprecation in latency is the result of redundant InterCloud application brokering which increased forwarding start time ϑ in resource layers $L = 3$ and $L = 4$ in comparison to $L = 2$. However, for resource layer $L = 1$, the increase in average latency is because of insufficient VM capacity. Intuitively speaking, when the scheduling policy for VMs in fog datacenters is *time-shared* and there is insufficient capacity, this may generally affect the average processing time. On the other hand, the average latency for high peak homogenous IoT application demand is reduced. Intuitively, capacity sufficiency can be attained as an increase in the size of resource layers increased InterCloud application brokering. This is because increasing VM capacity at the resource layers means there would be sufficient resources in the

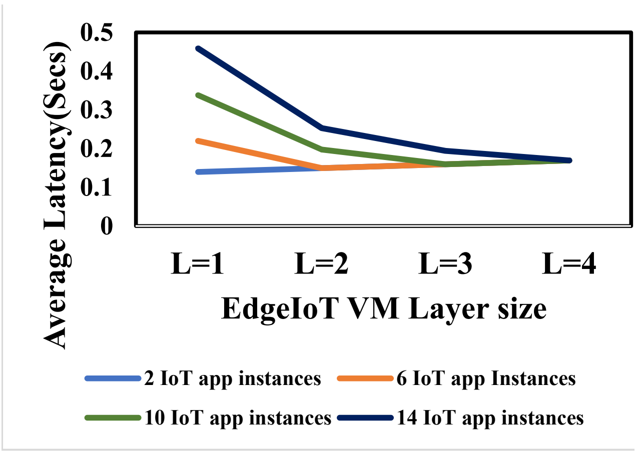


Fig.8. Average latency in seconds of SDN-based InterCloud application controllers for varying peak IoT application demand.

MEC environment for processing IoT application instances.

C. *Application Processing Cost*

In all the experiments, the cost of processing per second was set at 3.0 for each edgeIoT application instance. We compute the processing cost for each application instance as processing cost per second * processing time. For low peak application demands, *Fig.9* shows that the average cost of processing applications for resource layers is most minimised if there is sufficient VM capacity for processing IoT application demand and the resource layer size is sufficiently small. However, for high peak application demand, the average cost of processing is most reduced also if there is just enough capacity. This is more likely the case when the resource layer size is adequately large.

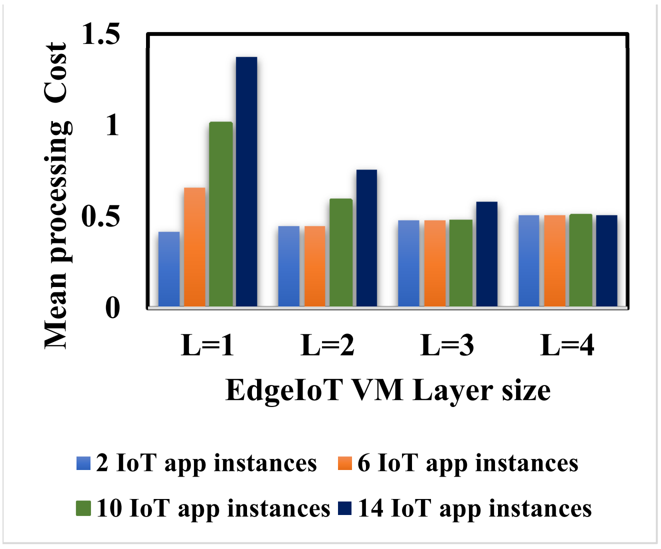


Fig.9. Average processing cost of SDN-based InterCloud application controllers for varying peak IoT application demand.

VII. CONCLUSION

Undoubtedly, MEC is becoming an important facilitator of consumer-centric IoT applications and services that demand real-time operation. It is important that schemes for managing resources guarantee QoS related to low latency in MEC or EdgeIoT platforms. As such, in this paper, we have proposed the architecture for adaptive QoS-aware capacity management in SDN-based InterCloud environments for MEC. This paper has provided the design of a self-adaptive capacity manager to ensure low latency for real-time edgeIoT applications. Furthermore, to adaptively improve QoS, the manager adopted for an implementation of the single node IoT router, an IP-based self-tuning minimum cost policy control plan. The outcome from experimental test-beds using CloudSim toolkit 4.0 have shown that the capacity management approach using dynamic programming and anytime optimisation can control QoS by applying policy decisions. However, our approach may be constrained by time considering large search spaces and highly dynamic application demand. It is for this reason the anytime algorithm was chosen, given that search can be terminated at any time before execution completes to return good or near-optimal solutions. Because of the exponential growth in the search space, it is also recommended that any policy control procedure be implemented as search using offline data for large scale SDN data planes. In the future, we would consider an application of coalition games with positive externalities using distributed SDN-based InterCloud application controllers as well as conducting more experiments and evaluation to extend this work further.

ACKNOWLEDGMENT

The first Author hereby gratefully acknowledges the financial support by Tertiary Education Trust Fund (TETFUND) / Abubakar Tafawa Balewa University, Nigeria to pursue doctorate degree program at De Montfort University Leicester, UK.

REFERENCES

[1] X. Sun, and N. Ansari (2016). EdgeIoT: Mobile Edge Computing for the Internet of Things. *IEEE Communications Magazine*, 2016, *54*(12), 22-29.

[2] T. Aoyama, and H. Sakai (2011) ‘Inter-cloud-computing’, *WIRTSCHAFTSINFORMATIK*, 53(3), pp. 171–175. doi: 10.1007/s11576-011-0272-4.

[3] R. N. Calheiros, A.N. Toosi, C. Vecchiola, and R. Buyya. A coordinator for scaling elastic applications across multiple clouds. *Future Generation Computer Systems*, 2012, *28*(8), 1350-1362.

[4] A. N. Toosi, R.N. Calheiros, and R. Buyya. Interconnected Cloud Computing Environments: Challenges, Taxonomy, and Survey. ACM Comput. Surv, 2014, 47(1).

[5] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge Computing: Vision and Challenges, 2016.

[6] E. Ahmed, and M.H. Rehmani, Mobile Edge Computing: Opportunities, solutions, and challenges. *Future Generation Computer Systems, 2016*

[7] P. Corcoran, and S.K. Datta. Mobile-edge computing and the internet of things for consumers: Extending cloud computing and services to the edge of the network. *IEEE Consumer Electronics Magazine*, 2016, *5*(4), 73-74.

[8] D. Bernstein, E. Ludvigson, K. Sankar, S. Diamond, and M. Morrow, Blueprint for the intercloud-protocols and formats for cloud computing interoperability. In *Internet and Web Applications and Services, 2009. ICIW'09. Fourth International Conference on* (pp. 328-336). IEEE.

[9] K.M. Sim. Agent-based Approaches for Intelligent InterCloud Resource Allocation. *IEEE Transactions on Cloud Computing, 2016.*

[10] E. Haleplidis, K., Pentikousis, S., Denazis, J. H., Salim, D., Meyer, and O., Koufopavlou. *Software-defined networking (SDN): Layers and architecture terminology*. No. RFC 7426. 2015.

[11] Fundation, Open Networking. "Software-defined networking: The new norm for networks." *ONF White Paper* 2 (2012): 2-6.

[12] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohmé. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 1999, *111*(1), 209-238.

[13] T. Rahwan, T. Michalak, M. Wooldridge, and N.R. Jennings. Anytime coalition structure generation in multi-agent systems with positive or negative externalities, 2012, *Artificial Intelligence*, *186*, 95-122.

[14] R. N. Calheiros, R. Ranjan, A. Beloglazov, C.A. De Rose, and R. Buyya. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 2011, *41*(1), 23-50.

[15] R. Buyya, R., Ranjan, and R.N., Calheiros. "Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities." In *High Performance Computing & Simulation, 2009. HPCS'09. International Conference on*, pp. 1-11. IEEE, 2009.

[16] R., Buyya, R., Ranjan, and R. N. Calheiros. "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services." In *International Conference on Algorithms and Architectures for Parallel Processing*, pp. 13-31. Springer Berlin Heidelberg, 2010

[17] D.C., Marinescu. Cloud computing: theory and practice. Newnes, 2013.

[18] A., Takefusa, H.,Nakada, R., Takano, T., Kudoh, and Y., Tanaka. "GridARS: A grid advanced resource management system framework for intercloud." In *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, pp. 705-710. IEEE, 2011.

[19] G.L. Stavrinides, and H.D. Karatza. "A cost-effective and QoS-aware approach to scheduling real-time workflow applications in PaaS and SaaS clouds." In *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, pp. 231-239. IEEE, 2015.

[20] M.C.,Chuang, M.C., Chen, and Y.H., Lin. "SDN-based resource allocation scheme in ultra-dense OFDMA smallcell networks." In *Advanced Materials for Science and Engineering (ICAMSE), International Conference on*, pp. 524-527. IEEE, 2016.

[21] W. Yasuhiro et al., “A Job Management Framework Enabling Deployment of Allocation Policy Regarding Computational and Network Resources,” IEICE TRANSACTIONS on Information and Systems, vol. J97D, no. 6, pp. 1082–1093, 2014.

[22] M., Shimizu, , Y., Watashiba, S., Date, and S., Shimojo. "Adaptive Network Resource Reallocation for Hot-Spot Avoidance on SDN-Based Cluster System." In *Cloud Computing Technology and Science (CloudCom), 2016 IEEE International Conference on*, pp. 608-613. IEEE, 2016

[23] L., Han, Z., Li, W., Liu, K., Dai, and W., Qu. "Minimum Control Latency of SDN Controller Placement." In *Trustcom/BigDataSE/I SPA, 2016 IEEE*, pp. 2175-2180. IEEE, 2016.

[24] M., Ojo, D., Adami, and S., Giordano. "A SDN-IoT Architecture with NFV Implementation." In *Globecom Workshops (GC Wkshps), 2016 IEEE*, pp. 1-6. IEEE, 2016.

[25] J., Li, J.H., Yoo, and J.W.K., Hong. "CPMan: Adaptive control plane management for software-defined networks." In *Network Function Virtualization and Software Defined Network (NFV-SDN), 2015 IEEE Conference on*, pp. 121-127. IEEE, 2015.

[26] C.,Yu, C.,Lumezanu, A., Sharma, Q., Xu, G., Jiang, and H.V., Madhyastha. "Software-defined latency monitoring in data center networks." In *International Conference on Passive and Active Network Measurement*, pp. 360-372. Springer International Publishing, 2015.

[27] D., Bernstein, and D., Vij. "Intercloud federation using via semantic resource federation API and dynamic SDN provisioning." In *Network of the Future (NOF), 2014 International Conference and Workshop on the*, pp. 1-8. IEEE, 2014.

[28] D.,Vij, Deepak, D., Bernstein, M., Morsey, P., Grosso, T., Magedanz, and A., Willner. "Software defined Bearer Intercloud networks semantic-based network exchange for the IEEE P2302 Intercloud approach." In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2015 12th International Conference on*, pp. 1-6. IEEE, 2015.

[29] M., Aazam, and E.N., Huh. "Cloud broker service-oriented resource management model." *Transactions on Emerging Telecommunications Technologies* (2015).

[30] R., Mehrotra, S., Srivastava, I., Banicescu, and S., Abdelwahed. "Towards an autonomic performance management approach for a cloud broker environment using a decomposition–coordination based methodology." *Future Generation Computer Systems* 54 (2016): 195-205.

[31] F., Jrad, J. Tao, and A., Streit. "SLA based Service Brokering in Intercloud Environments." *CLOSER* 2012 (2012): 76-81.

[32] A.A., Dixit, F., Hao, S. Mukherjee, T. V. Lakshman, and R., Kompella. "Elasticon: an elastic distributed sdn controller." In *Proceedings of the tenth ACM/IEEE symposium on Architectures for networking and communications systems*, pp. 17-28. ACM, 2014.

[33] R., Mijumbi, J., Serrat, J., Rubio-Loyola, N., Bouten, F., De Turck, and S., Latré. "Dynamic resource management in sdn-based virtualized networks." In *Network and Service Management (CNSM), 2014 10th International Conference on*, pp. 412-417. IEEE, 2014.

[34] D., Ardagna, S., Casolari, M., Colajanni, and B., Panicucci. "Dual time-scale distributed capacity allocation and load redirect algorithms for cloud systems." *Journal of Parallel and Distributed Computing* 72, no. 6 (2012): 796-808.

[35] A., Amokrane, J.Hwang, J., Xiao, and A., Nikos. "Dynamic capacity management and traffic steering in enterprise passive optical networks." In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pp. 406-413. IEEE, 2015.

[36] D., Adami, L., Donatini, S., Giordano, and M., Pagano. "A network control application enabling software-defined quality of service." In *Communications (ICC), 2015 IEEE International Conference on*, pp. 6074-6079. IEEE, 2015.

[37] M.F., Bari, S. R. Chowdhury, R., Ahmed, and R., Boutaba. "PolicyCop: An autonomic QoS policy enforcement framework for software defined networks." In *Future Networks and Services (SDN4FNS), 2013 IEEE SDN For*, pp. 1-7. IEEE, 2013.

[38] K .M. Sim. Agent-based interactions and economic encounters in an intelligent InterCloud. *IEEE Transactions on Cloud Computing*, 2015, *3*(3), 358-371.

[39] T. Rahwan, T.P. Michalak, M. Wooldridge, N.R. Jennings. Coalition Structure Generation: A Survey. Artificial Intelligence, 2015, 229, pp 139-174.

[40] I.E. Hafalir. "Efficiency in coalition games with externalities." *Games and Economic Behavior* 61, no. 2 (2007): 242-258.

[41] N. Grozev, and R. Buyya. Inter-Cloud architectures and application brokering: taxonomy and survey. *Software: Practice and Experience*, 2014, *44*(3), 369-390.

[42] P.T. Endo et al. Resource allocation for distributed cloud: concepts and research challenges. *IEEE network*, 2011, *25*(4), 42-46.

[43] J. Weinman. Axiomatic cloud theory. Online (2011), http://www. joeweinman. com/Resources/Joe_Weinman_Axiomatic_Cloud_Theory. pdf.

[44] T. Rahwan, S.D. Ramchurn, V.D. Dang, A. Giovannucci, and N.R. Jennings. Anytime optimal coalition structure generation, 2007, In *AAAI*, 7, pp. 1184-1190.

[45] S.O. *Aliyu, F. Chen, & H. Li, A Self-Tuning Procedure for Resource Management in InterCloud Computing. In Software Quality, Reliability and Security Companion (QRS-C), 2016 IEEE International Conference on (pp. 326-333). IEEE.*

[46] K.W., Jee, D.L., McShan, and B.A., Fraass. "Lexicographic ordering: intuitive multicriteria optimization for IMRT." *Physics in Medicine and Biology* 52, no. 7 (2007): 1845

[47] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, 2009, *8*(4), 14-23