UNIVERSITY OF NOVI SAD

FACULTY OF TECHNICAL SCIENCES
NOVI SAD

Miloš Simić

Micro clouds and edge computing as a service

- Ph. D. Thesis -

Supervisor
Goran Sladić, PhD, associate professor

Novi Sad, 2021.

Miloš Simić: *Micro clouds and edge computing as a service*

SERBIAN TITLE: Micro clouds and edge computing as a service

SUPERVISOR:
Goran Sladić, PhD, associate professor

LOCATION:
Novi Sad, Serbia

DATE:
September 2021

UNIVERZITET U NOVOM SADU • **FAKULTET TEHNIČKIH NAUKA**
21000 NOVI SAD, Trg Dositeja Obradoviće 6

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Redni broj, **RBR**:	
Identifikacioni broj, **IBR**:	
Tip dokumentacije, **TD**:	Monografska dokumentacija
Tip zapisa, **TZ**:	Tekstualni štampani materijal
Vrsta rada, **VR**:	Doktorska disertacija
Autor, **AU**:	Miloš Simić
Mentor, **MH**:	dr Goran Sladić, vanredni profesor
Naslov rada, **NR**:	Micro clouds and edge computing as a service
Jezik publikacije, **JP**:	engleski
Jezik izvoda, **JI**:	srpski
Zemlja publikacije, **ZP**:	Srbija
Uže geografsko područije, **UGP**:	Vojvodina
Godina, **GO**:	2021
Izdavač, **IZ**:	Fakultet tehničkih nauka
Mesto i adresa, **MA**:	Trg Dositeja Obradovića 6, 21000 Novi Sad
Fizički opis rada, **FO**: (poglavlja/strana /citata/tabela/slika/grafika/priloga)	6/159/154/13/13/0/0
Naučna oblast, **NO**:	Elektrotehničko i računarsko inženjerstvo
Naučna disciplina, **ND**:	Distribuirani sistemi
Predmetna odrednica/Ključne reči, **PO**:	distributed systems, cloud computing, microservices, software as a service, edge computing, micro clouds
UDK	
Čuva se, **ČU**:	Biblioteka Fakulteta tehničkih nauka, Trg Dositeraj Obradovića 6, 21000 Novi Sad
Važna napomena, **VN**:	

UNIVERZITET U NOVOM SADU • **FAKULTET TEHNIČKIH NAUKA**
21000 NOVI SAD, Trg Dositeja Obradoviće 6

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Izvod, **IZ**:		U sklopu disertacije izvršeno je istraživanje u oblasti razvoja bezbednog softvera. Razvijene su dve metode koje zajedno omogućuju integraciju bezbednosne analize dizajna softvera u proces agilnog razvoja. Prvi metod predstavlja radni okvir za konstruisanje radionica čija svrha je obuka inženjera softvera kako da sprovode bezbednosnu analizu dizajna. Drugi metod je proces koji proširuje metod bezbednosne analize dizajna kako bi podržao bolju integraciju spram potreba organizacije. Prvi metod je evaluiran kroz kontrolisan eksperiment, dok je drugi metod evaluiran upotrebom komparativne analize i analize studija slučaja, gde je proces implementiran u kontekstu dve organizacije koje se bave razvojem softvera.	
Datum prihvatanja teme, **DP**:			
Datum odbrane, **DO**:			
Članovi komisije, **KO**:	Predsednik:	dr Branko Milosavljević, redovni profesor, FTN, Novi Sad	
	Član:	dr Silvia Gilezan, redovni profesor, FTN, Novi Sad	
	Član:	dr Gordana Milosavljević, vanredni profesor, FTN, Novi Sad	Potpis mentora
	Član:	dr Žarko Stanisavljević, docent, ETF, Beograd	
	Član, mentor:	dr Goran Sladić, vanredni profesor, FTN, Novi Sad	

UNIVERSITY OF NOVI SAD • **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:	
Identification number, **INO**:	
Document type, **DT**:	Monograph documentation
Type of record, **TR**:	Textual printed material
Contents code, **CC**:	Ph.D. thesis
Author, **AU**:	Miloš Simić
Mentor, **MN**:	Goran Sladić, Ph.D., Associate Professor
Title, **TI**:	Micro clouds and edge computing as a service
Language of text, **LT**:	English
Language of abstract, **LA**:	Serbian
Country of publication, **CP**:	Serbia
Locality of publication, **LP**:	Vojvodina
Publication year, **PY**:	2021
Publisher, **PB**:	Faculty of Technical Sciences
Publication place, **PP**:	Trg Dositeja Obradovića 6, 21000 Novi Sad
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/	6/159/154/13/13/0/0
Scientific field, **SF**:	Electrical engineering and computing
Scientific discipline, **SD**:	Distributed systems
Subject/Key words, **S/KW**:	distributed systems, cloud computing, microservices, software as a service, edge computing, micro clouds
UC	
Holding data, **HD**:	Library of Faculty of Technical Sciences, Trg Dositeja Obradovića 6, 21000
Note, **N**:	

UNIVERSITY OF NOVI SAD • **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Abstract, **AB**:		This thesis presents research in the field of secure software engineering. Two methods are developed that, when combined, facilitate the integration of software security design analysis into the agile development workflow. The first method is a training framework for creating workshops aimed at teaching software engineers on how to perform security design analysis. The second method is a process that expands on the security design analysis method to facilitate better integration with the needs of the organization. The first method is evaluated through a controlled experiment, while the second method is evaluated through comparative analysis and case study analysis, where the process is tailored and implemented for two different software vendors.	
Accepted by the Scientific Board on, **ASB**:		11.07.2019.	
Defended on, **DE**:			
Defended Board, **DB**:	President:	Branko Milosavljević, PhD, Full Professor, FTN, Novi Sad	
	Member:	Silvia Gilezan, PhD, Full Professor, FTN, Novi Sad	
	Member:	Gordana Milosavljević, PhD, Associate Professor, FTN, Novi Sad	Menthor's signature
	Member:	Žarko Stanisavljević, PhD, Assistant Professor, ETF, Belgrade	
	Member, Mentor:	Goran Sladić, PhD, Associate Professor, FTN, Novi Sad	

Acknowledgements

First of all, I would like to express my sincere gratitude to my mentors Professor Jovanka Pantović and Professor Hugo Torres Vieira, for their unselfish help and experienced guidance. Before starting my PhD studies I had no idea what doing research means. They introduced me to the wonderful world of formal methods and shown me how far one needs to go in order to extract the juice out of his work. Above all, I thank them for always encouraging me to "keep up the good work", and for demonstrating what it takes to be a true scientist and a good person.

To my elementary and high school teachers, and my professors at the university, thank you for making me prepared for doing the Ph.D.

I would like to thank my colleagues at the Chair of Mathematics at Faculty of Technical Sciences for letting me be part of the family and for sharing all the good and the bad that our job carries.

To all of my friends, for making my life happier.

To my family, I owe the most. I thank my "in-law" family Radovan, Jasna, and Maja, for their great support. To my sister Draginja, her husband Bojan and my nieces Jelisaveta and Jovana, to my older brother Mladen and his wife Nina, and to my younger brother Aleksandar, thank you for being a perfect family. To my parents Jovanka and Siniša, for their endless love. To my wife Sanja, and our daughters Tamara and Lenka, for their patience, support, and unquestionable love, thank you for being my world.

Abstract

Distributed software systems have changed the way people communicate, learn and run businesses: almost all aspects of human life have become connected to the internet. The system of interconnected computing devices has numerous positive impacts on everyday life, however, it also raises some concerns, among which are security, accessibility and availability issues.

This thesis investigates problems of formal, mathematically based, representation and analysis of controlled usage and sharing of resources in distributed software systems. The thesis is organized into four chapters. The first chapter provides motivation for our work, and the last concludes the thesis. The second and the third chapters are the core of the thesis, the former addresses controlling information passing and the latter addresses controlling usages of resources.

The second chapter presents a model for confidential name passing, called Confidential π-calculus, abbreviated C_π. This model is a simple fragment of the π-calculus that disables information forwarding directly at the syntax level. We provide an initial investigation of the model by presenting some of its properties, such as the non-forwarding property and the creation of closed domains for channels. We also present examples showing that C_π can be used to model restricted information passing, authentication, closed and open-ended groups. We present an encoding of the (sum-free) π-calculus in C_π and we prove the correctness of the encoding via an operational correspondence result.

The third chapter presents a model of floating authorizations. Our process model introduces floating authorizations as first-class entities, encompassing dimensions of accounting, domain, and delegation. We exploit the language of an already existing process algebra for authorizations, and we adopt a different semantic interpretation so as to capture accounting. We define the semantics of our model in two equivalent ways, using a labeled transition system and a reduction relation. We define error processes as undesired configurations that cannot evolve due to lacking authorizations. The thesis also provides a preliminary investigation of the behavioral semantics of our authorization model, showing some fundamental properties and also informing on the specific nature of floating authorizations.

In the third chapter, we also present a typing discipline that allows to statically single out processes that are not errors and that never evolve into errors, addressing configurations where authorization assignment is not statically prescribed in the system specification. We also develop a refinement of our typing discipline to pave the way for a more efficient type-checking procedure. We show an extended example of a scenario that involves the notion of Bring Your Own License, and we exploit this example to provide insight on a possible application of our model in programming language design.

Key words: distributed systms, cloud computing, micro clouds, edge computing

Rezime

Ključ reči: distributed systms, cloud computing, micro clouds, edge computing

Table of Contents

List of Figures

List of Tables

List of Abbreviations

CC	Cloud computing
AWS	Amazon Web Services
IoT	Internet of Things
DS	Distributed systems
DC	Data center
IaaS	infrastructure as a service
PaaS	Platform as a service
SaaS	Software as a service
CaaS	Container as a service
DBaaS	Databae as a service
XaaS	Everything as a Service
P2P	Peer-to-peer
DHT	Distributed Hash Table
NoSQL	Not Only SQL
EC	Edge computing

MEC Mobile edge computing

MCC Mobile cloud computing

QoE Quality of experience

Chapter 1

Introduction

Various software systems has changed the way people communicate, learn and run businesses, and interconnected computing devices has numerous positive applications in everyday life. Over the past decade, computation and data volumes have increased significantly [8]. Augmented reality, online gaming, face recognition, autonomous vehicles, or the Internet of Things (IoT) applications produce huge volumes of data. Workloads like those require latency below a few tens of milliseconds [8]. These requirements are outside what a centralized model like the CC can offer [8]. Even small problems can contribute to large downtime of applications and servies people are depending on. Recent example is yet another outage that happend in Amazon Web Services (AWS), and as a result a large amount of intrnet become unavaleble.

The aim of this thesis is to provide formal models uppon which we implement distributed system for oragnizing cloud-like geo-distributed environments for users or CC providers to utilize, in order to minimize downtime of critical services. The whole system can be looked as a pre-cloud or pre-processing layer sending only important data to the cloud minimizing cost for users, and ensuring avelability of CC services. Ensuring reliability and correctness of any system is very difficult, and should be mathematically based. Formal methods are techniques

that allow us specification and verification of complex (software and hardware) systems based on mathematics and formal logic.

We start by describing the general problem area that our work addresses in Section 1.1. Sections 1.2 and 1.3 describe the theoretical background behind the problem, where we examine distributed systems and distributed computing, focusing on design details, communication patters and organizational structure. In Section 1.4, we specify the exact problem that our work addresses and describe our hypothesis and research goals in Section 1.5. Section 1.6 present the structure of the thesis.

1.1 Problem area

Cloud centralized architecture with enormous data-centers (DCs) capacities creates an effective economy of scale to lower administration cost [3]. However, when such a system grows to its limits, centralization brings more problems than solutions [14, 22]. Despite all the CC benefits, applications and services face a serious degradation over time, due to the high bandwidth and latency [18]. This can have a huge consequence on the business and potentially human lives as well. Organizations use cloud services to avoid huge investments [24], like creating and maintaining their own DCs. They consume resources created by others [26] and pay for usage time – a pay as you go model.

Data is required to be moved to the cloud from data sources, which introduces a high latency in the system [16]. For example, Boeing 787s generates half a terabyte of data per single flight, while a self-driving car generates two petabytes of data per single drive. Bandwidth is not large enough to support such requirements [6]. Data transfer is not the only problem: applications like self-driving cars, delivery drones, or power balancing in electric grids require real-time processing for proper decision making [6]. We might face serious issues if a cloud service becomes unavailable due to denial-of-service attack, network, or cloud failure [14].

To overcome cloud latency, research led to new computing areas, and model in which computing and storage utilities are in proximity to data sources [26]. The cloud is enhanced with new ideas for future generation applications [25].

1.2 Distributed systems

There are various definitions of distributed systems (DS), but we can think of DS as a systems where multiple entities can comunicate to one another in some way, but at the same time, they are able to performing some operations. Three significant characteristics of distributed systems are: (1) **concurrency of components**, (2) **lack of a global clock**, and (3) **independent failure of components** [30]. In [31] authors give formal definition "distributed system is a collection of autonomous computing elements that appears to its users as a single coherent system".

When talking about DS, we usually think about computing systems that are connected via network or over the internet. But DS are not exclusiv to domain of cumputer science. They existed before computers started to enrich almost every aspect of human life. DS have been used in varios different domains such as: **telecommunication networks**, **aircraft control systems**, **industrial control systems** etc. DS are used anywhere where amout of users are growing rapidly, so that single entity can't reponse to users demands in (near) real-time.

This section will explain different aspects of DS in computing systems, that are important for future pars of the thesis. Section 1.2.1 gives more details about scalability and what it meens in modern day computer applications. Section 1.2.2 gives explanation what CC is, organizational aspects of CC as well as used models. Section 1.2.3 gives explanation what peer-to-peer networks are, and why are they important in modern DS. Section 1.2.4 gives general deffinition what mobile computing is and new ways of implementation DS.

1.2.1 Scalability

Scalability is the property of a system to handle a growing amount of work by adding resources to the system [4]. When talking about computer systems scalability can be represented in two flavors:

- **Scaling vertically** means upgradeing the hardware that computer systems are running on. Vertical scaling can increase performance to what latest hardware can offer, and here we are limited by the laws of phisics and Moor's law [15]. Tipical example that require this type of scaleing is relation dataase server. These capabilities are insufficient for moderate to big workloads.
- **Scaling horizontally** means that we scale our system by keep adding more and more computers, rather than upgrading the hardware of a single one. With this apporouch we are (almost) limitless how much we can scale. Whenever performance degrades we can simply add more computers (or nodes). These nodes are not required to be some high-end machines.

Table 1.1 summarize differences between horizontall and verticall scaling.

	Scaling vertically	**Scaling horizontally**
Scaling	Limited	Unlimited
Managment	Easy	Comlex
Investments	Expensive	Afordable

Table 1.1: Differences between horizontall and verticall scaling.

Scaling horizontally is a preferable way for scaling DS, not because we can scale easier, or because it is significantly cheaper than vertical scaling after a certain threshold [4] but because this approuch comes with few more benefits that are esspecially important when talking large-scale DS. Adding more nodes gives us two important properties:

- **Fault tolerance** means that applications running on multiple places at the same time, are not bound to the fail of node, cluster or even data center (DC). As long as there is a copy of application running somewhere, user will get response back. This means that service is more **avalible**, that running on a single node no metter how high-end that node is. Eventually all nodes are going to break.

- **Low latency** refers to the idea that the world is limited by the speed of light. If a node running application is too far away, user will wait too long for the response to get back. If same application is running on multiple places, user request will hit node that is closest to the user.

But despite all the obvious benefits, for a DS to work properly, we need the write software in such a way that is able to run on multiple nodes, as well as that accept **failure** and deal with it. This turns out to be not an easy task.

1.2.2 Cloud computing

We can define cloud computing (CC) like aggregation of computing resources as a utility, and software as a service [32]. Hardware and software in big DC provide services for user consumption over the internet [1]. Resources like CPU, GPU, storage, and network are utilities and can be used as well as released on-demand [35]. The key strength of the CC are offered services [32].

The traditional CC model provides enormous computing and storage resources elastically, to support the various applications needs. This property refers to the cloud ability to allow services, allocation of additional resources, or release unused ones to match the application workloads on-demand [10]. Services usually fall in one of three main categories:

- **Infrastructure as a service (IaaS)** allows businesses to purchase resources on-demand and as-needed instead of bying and manageing hardware themself.
- **Platform as a service (PaaS)** delivers a framework for developers to create, maintain and manage their applications. All resources are managed by the enterprise or a third-party vendor.
- **Software as a service (SaaS)** deliver applications over the internet to its users. These applications are managed by a third-party vendor, .

Figure 1.1 show difference in control and management of resources between different cloud options and on-premises solution.

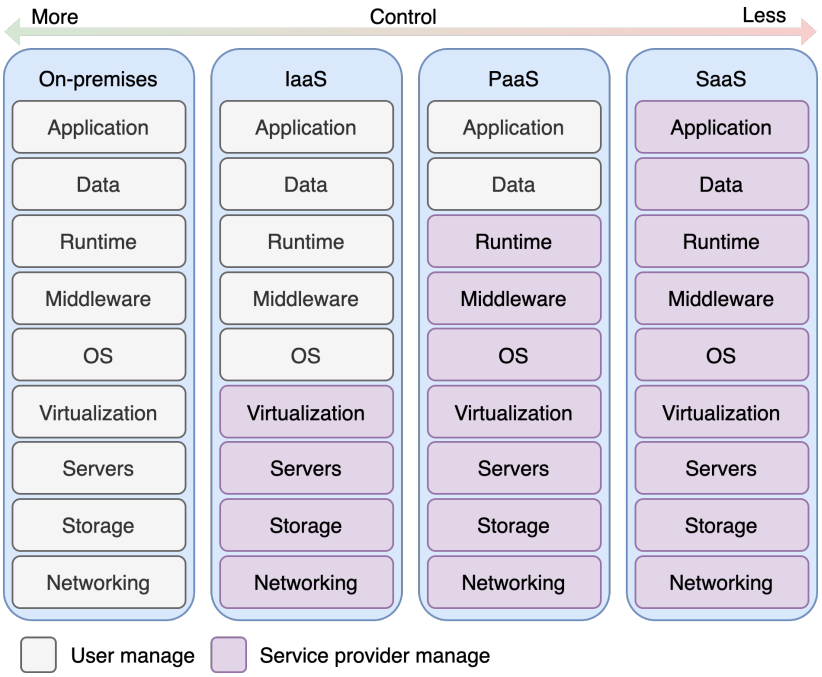

Figure 1.1: Difference between cloud options and on-premises solution.

The user can choose a single solution, or combine more of them if such a thing is required depending on preferences and needs.

CC has been the dominating tool in the past decade in various applications [26]. It is changing, evolving, and offering new types of services. Resources such as container as a service (CaaS), database as a service (DBaaS) [23] are newly introduced. The CC model gives us a few benefits. Centralization relies on the economy of scale to lower the cost of administration of big DC. Organizations using cloud services avoid huge investments. Like creating and maintaining their own DC. They consume resources created by others [26] and pay for usage time – a pay as you go model.

Over the years there are more as a service options avalible, forming **everything as a service (XaaS)** model [11]. This model propose that any hardware or software resource can be ofered as a service to the users over the internet.

Table 1.2 shows common examples of SaaS, PaaS, and IaaS applications.

Platform type	**Common Examples**
IaaS	AWS, Microsoft Azure, Google Compute Engine
PaaS	AWS Elastic Beanstalk, Azure, App Engine
SaaS	Gmail, Dropbox, Salesforce, GoToMeeting

Table 1.2: common examples of SaaS, PaaS, and IaaS.

CC is giving a user an illusion that he is using single machine, while the backgroud implementaion is fairly complicated and consists of various elements that are composed of countles machines. CC is tipical example of horizontally scalable system explained in 1.2.1

1.2.3 Peer-to-peer networks

Peer-to-peer (P2P) communication is a networking architecture model that partitions tasks or workloads between peers [28]. All peers are

created equally in the system, and there is no such thing as a node that is more important then others. Every Peer have a portion system resources, such as processing power, disk storage or network bandwidth, directly available to other network participants, without the need for central coordination by servers or stable hosts [28]. P2P nodes are connected and share resources without going through a separate server computer that is responsabile for routing. Figure 1.2 show difference in network topology between P2P networks (a) and client-server architecture (b).

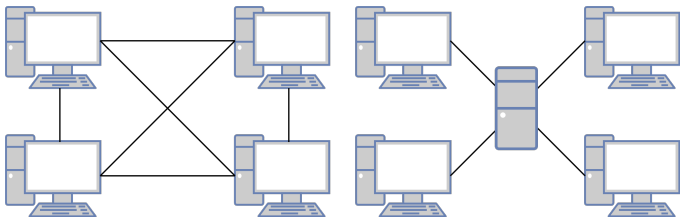

Figure 1.2: (a) P2P network and (b) client-server network.

Peers are creating a sense of virtual community. This community of peers can resolve a greater tasks, beyond those that individual peers can do. Yet, these tasks are beneficial to all the peers in the system [2].

Based on how the nodes are linked to each other within the overlay network, and how resources are indexed and located, we can classify networks as [17]:

- **Unstructured** do not have a particular structure by design, but they are formed by nodes that randomly form connections [13]. Their strenght and weaknes at the same time is ther lack of structure. These networks and robust when peers join and leave network. But when doing query, they must found more possible peers that have same peace of data. Tipical example of this group is a Gossip-based protocols like [9].

- **Structured** peers are organized into a specific topology, and the protocol ensures that any node can efficiently search the network

for a resource. The famos type of structured P2P network is a Distributed Hash Table (DHT). These networks maintain lists of neighbors to do more efficent lookup, and as such they are not so robust when nodes join or leave the network. DHT commonly used in resource loopkup systems [29], and as efficent resource lookip management and scheduling of applications, or as an integral part of distributed storage systems and NoSQL[20] databases.

- **Hybrid** combine previous two models in various ways.

P2P networks are great tool in many arsenals, but because their unique ability to act as a server and as a client at the same time we must be aware and pay more attention to security because they are more vulnerable to exploits [33]/

1.2.4 Mobile computing

Mobile cloud computing (MCC), was the first idea that introduced task offloading [12, 21]. Heavy computation remains in the cloud. Mobile devices run small client software and interact with the cloud, over the internet using his resources.

The main problem with MCC is that the cloud is usually far away from end devices. That leads to high latency and bad quality of experience (QoE) [21]. Especially for latency-sensitive applications. Even though MCC is not that much different from the standard cloud model. We had moved a small number of tasks from the cloud. Thus opening the door for future models.

To overcome cloud latency and MCC problems, research led to new computing areas like edge computing (EC). EC is a model in which computing and storage utilities are in proximity to data sources [26]. The cloud is enhanced with new ideas for future generation applications [25].

Over the years, designs like fog [5], cloudlets [24], and mobile edge computing (MEC) [34] emerged. In this thesis, we refer to all these

models as edge nodes. They all use the concept of data and computation offloading from the cloud closer to the ground [19], while heavy computation remains in the cloud because of resource availability [25].

EC models introduce small-scale servers that operate between data sources and the cloud. Typically, they have much less capabilities compared to the cloud counterparts [7]. These servers can be spread in base stations [34], coffee shops, or over geographic regions to avoid latency as well as huge bandwidth [24]. They can serve as firewalls [27] and pre-processing tier, while users get a unique ability to dynamically and selectively control the information sent to the cloud.

1.3 Distributed computing

1.3.1 Big Data

1.3.2 Microservices

1.3.3 Parallel computing

1.4 Problem Statement

1.5 Research Hypotheses, and Goalst

1.6 Structure of the thesis

Chapter 2

Research review

2.1 Platform models

2.2 Nodes organization

2.3 Task offloading

Chapter 3

Micro clouds and edge computing as a service

3.1 Formal model

3.1.1 Multiparty asynchronous session types

3.1.2 Health-check protocol

3.1.3 Cluster formation protocol

3.1.4 List detail protocol

3.2 Configurable Model Structure

3.3 Applications Model

3.3.1 Packaging

3.3.2 Execution

3.4 Separation of concers

3.5 As a service model

3.6 Results

Chapter 4

Conclusion

4.1 Summary of contributions

4.2 Future work

Bibliography

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the clouds: A berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[2] H. M. N. D. Bandara and A. P. Jayasumana. Collaborative applications over peer-to-peer systems-challenges and solutions. *Peer Peer Netw. Appl.*, 6(3):257–276, 2013.

[3] M. F. Bari, R. Boutaba, R. P. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang, and M. F. Zhani. Data center network virtualization: A survey. *IEEE Commun. Surv. Tutorials*, 15(2):909–928, 2013.

[4] A. B. Bondi. Characteristics of scalability and their impact on performance. In *Second International Workshop on Software and Performance, WOSP 2000, Ottawa, Canada, September 17-20, 2000*, pages 195–203. ACM, 2000.

[5] F. Bonomi, R. A. Milito, P. Natarajan, and J. Zhu. Fog computing: A platform for internet of things and analytics. In N. Bessis and C. Dobre, editors, *Big Data and Internet of Things: A Roadmap for Smart Environments*, volume 546 of *Studies in Computational Intelligence*, pages 169–186. Springer, 2014.

[6] J. Cao, Q. Zhang, and W. Shi. *Edge Computing: A Primer*. Springer Briefs in Computer Science. Springer, 2018.

[7] M. Chen, Y. Hao, Y. Li, C. Lai, and D. Wu. On the computation offloading at ad hoc cloudlet: architecture and service modes. *IEEE Commun. Mag.*, 53(6-Supplement):18–24, 2015.

[8] M. Chiang and T. Zhang. Fog and iot: An overview of research opportunities. *IEEE Internet Things J.*, 3(6):854–864, 2016.

[9] A. Das, I. Gupta, and A. Motivala. SWIM: scalable weakly-consistent infection-style process group membership protocol. In *2002 International Conference on Dependable Systems and Networks (DSN 2002), 23-26 June 2002, Bethesda, MD, USA, Proceedings*, pages 303–312. IEEE Computer Society, 2002.

[10] M. D. de Assunção, A. D. S. Veith, and R. Buyya. Distributed data stream processing and edge computing: A survey on resource elasticity and future directions. *J. Netw. Comput. Appl.*, 103:1–17, 2018.

[11] Y. Duan, G. Fu, N. Zhou, X. Sun, N. C. Narendra, and B. Hu. Everything as a service (xaas) on the cloud: Origins, current and future trends. In C. Pu and A. Mohindra, editors, *8th IEEE International Conference on Cloud Computing, CLOUD 2015, New York City, NY, USA, June 27 - July 2, 2015*, pages 621–628. IEEE Computer Society, 2015.

[12] N. Fernando, S. W. Loke, and J. W. Rahayu. Mobile cloud computing: A survey. *Future Gener. Comput. Syst.*, 29(1):84–106, 2013.

[13] I. Filali, F. Bongiovanni, F. Huet, and F. Baude. A survey of structured P2P systems for RDF data storage and retrieval. *Trans. Large Scale Data Knowl. Centered Syst.*, 3:20–55, 2011.

[14] H. S. Gunawi, M. Hao, R. O. Suminto, A. Laksono, A. D. Satria, J. Adityatama, and K. J. Eliazar. Why does the cloud stop computing? lessons from hundreds of service outages. In M. K. Aguilera, B. Cooper, and Y. Diao, editors, *Proceedings of the Seventh ACM Symposium on Cloud Computing, Santa Clara, CA, USA, October 5-7, 2016*, pages 1–16. ACM, 2016.

[15] J. L. Gustafson. *Moore's Law*, pages 1177–1184. Springer US, Boston, MA, 2011.

[16] S. K. A. Hossain, M. A. Rahman, and M. A. Hossain. Edge computing framework for enabling situation awareness in iot based smart city. *J. Parallel Distributed Comput.*, 122:226–237, 2018.

[17] M. Kamel, C. M. Scoglio, and T. Easton. Optimal topology design for overlay networks. In I. F. Akyildiz, R. Sivakumar, E. Ekici, J. C. de Oliveira, and J. McNair, editors, *NETWORKING 2007. Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet, 6th International IFIP-TC6 Networking Conference, Atlanta, GA, USA, May 14-18, 2007, Proceedings*, volume 4479 of *Lecture Notes in Computer Science*, pages 714–725. Springer, 2007.

[18] M. B. A. Karim, B. I. Ismail, M. Wong, E. M. Goortani, S. Setapa, L. J. Yuan, and H. Ong. Extending cloud resources to the edge: Possible scenarios, challenges, and experiments. In *International Conference on Cloud Computing Research and Innovations, ICCCRI 2016, Singapore, Singapore, May 4-5, 2016*, pages 78–85. IEEE Computer Society, 2016.

[19] A. Khune and S. Pasricha. Mobile network-aware middleware framework for cloud offloading: Using reinforcement learning to make reward-based decisions in smartphone applications. *IEEE Consumer Electron. Mag.*, 8(1):42–48, 2019.

[20] N. Leavitt. Will nosql databases live up to their promise? *Computer*, 43(2):12–14, 2010.

[21] L. Lin, X. Liao, H. Jin, and P. Li. Computation offloading toward edge computing. *Proc. IEEE*, 107(8):1584–1607, 2019.

[22] P. G. López, A. Montresor, D. H. J. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. P. Barcellos, P. Felber, and E. Rivière. Edge-centric computing: Vision and challenges. *Comput. Commun. Rev.*, 45(5):37–42, 2015.

[23] P. M. Mell and T. Grance. Sp 800-145. the nist definition of cloud computing. Technical report, Gaithersburg, MD, USA, 2011.

[24] S. A. Monsalve, F. G. Carballeira, and A. Calderón. A heterogeneous mobile cloud computing model for hybrid clouds. *Future Gener. Comput. Syst.*, 87:651–666, 2018.

[25] H. Ning, Y. Li, F. Shi, and L. T. Yang. Heterogeneous edge computing open platforms and tools for internet of things. *Future Gener. Comput. Syst.*, 106:67–76, 2020.

[26] M. Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, 2017.

[27] M. Satyanarayanan, G. Klas, M. D. Silva, and S. Mangiante. The seminal role of edge-native applications. In E. Bertino, C. K. Chang, P. Chen, E. Damiani, M. Goul, and K. Oyama, editors, *3rd IEEE International Conference on Edge Computing, EDGE 2019, Milan, Italy, July 8-13, 2019*, pages 33–40. IEEE, 2019.

[28] R. Schollmeier. A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications. In R. L. Graham and N. Shahmehri, editors, *1st International Conference on Peer-to-Peer Computing (P2P 2001), 27-29 August 2001, Linköping, Sweden*, pages 101–102. IEEE Computer Society, 2001.

[29] I. Stoica, R. T. Morris, D. R. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In R. L. Cruz and G. Varghese, editors, *Proceedings of the ACM SIGCOMM 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 27-31, 2001, San Diego, CA, USA*, pages 149–160. ACM, 2001.

[30] A. S. Tanenbaum and M. van Steen. *Distributed systems - principles and paradigms, 2nd Edition.* Pearson Education, 2007.

[31] M. van Steen and A. S. Tanenbaum. A brief introduction to distributed systems. *Computing*, 98(10):967–1009, 2016.

[32] W. Vogels. A head in the clouds the power of infrastructure as a service. In *Proceedings of the 1st Workshop on Cloud Computing and Applications*, 2008.

[33] Q. H. Vu, M. Lupu, and B. C. Ooi. *Peer-to-Peer Computing - Principles and Applications.* Springer, 2010.

[34] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang. A survey on mobile edge networks: Convergence of computing, caching and communications. *IEEE Access*, 5:6757–6779, 2017.

[35] Q. Zhang, L. Cheng, and R. Boutaba. Cloud computing: state-of-the-art and research challenges. *J. Internet Serv. Appl.*, 1(1):7–18, 2010.