PROCEEDINGS

The 3rd International Workshop

Dew Computing

(DEWCOM 2018)

Editors: Yingwei Wang and Karolj Skala

Toronto Canada
29th-30th October 2018

PATRONAGE

IEEE STC DEWCom

Table of Contents

Preface

This volume contains the papers presented at DEWCOM 2018: The 3rd International Workshop on Dew Computing held on October 29-30, 2018 in Toronto, Canada.

DEWCOM is an annual international workshop on dew computing. The first one, DEWCOM 2016, was held in Charlottetown, Canada. The second one, DEWCOM 2017, was held in Opatija, Croatia. DEWCOM 2018 was the third one in this series, and it was held together with the 28th Annual International Conference on Computer Science and Software Engineering (CASCON 2018).

Dew computing is a new post-cloud computing model appeared in 2015. While cloud computing uses centralized servers to provide various services, dew computing uses on-premises computers to provide decentralized, cloud-friendly, and collaborative micro services to end-users.

Dew computing is an on-premises computer software-hardware organization paradigm in the cloud computing environment, which does not contradict with cloud computing, does not replace cloud computing, but it is complementary to cloud computing. The key features of dew computing are that on-premises computers provide functionality independent of cloud services and they also collaborate with cloud services. Briefly speaking, dew computing is an organized way of using local computers in the age of cloud computing.

DEWCOM 2018 was organized by IEEE Computer Society Dew Computing Special Technical Community (DewCom STC). Currently, DewCom STC members are spread in 19 different countries: Canada, USA, Croatia, Austria, Germany, Romania, Sweden, Ukraine, United Arab Emirates, United Kingdom, Argentina, Colombia, El Salvador, China, India, Pakistan, Australia, Macedonia, and Nepal. More and more people are joining the dew computing community to get involved in research and development activities.

Each submission to DEWCOM 2018 was peer-reviewed by two or three reviewers. The committee decided to accept 8 papers. This proceedings included 7 of the accepted papers. These papers can be classified into the following three categories: dew computing models, dew computing formal descriptions, and dew computing applications.

In the dew computing models category, three papers were included in this proceedings. "Overview of Cloudlet, Fog Computing, Edge Computing, and Dew Computing" by Pan, Thulasiraman and Wang and "Edge and Dew Computing for Streaming IoT" by Gusev provided some viewpoints to observe the features and differences among post-cloud computing models. "The Rainbow Global Service Ecosystem" by Skala and Sojat discussed the future forms of computing organizations.

Dew computing formal descriptions is a new category. "Formal Description of Dew Computing" by Gusev and Wang proposed a formal description of dew computing. It could be helpful for researchers and professionals to get a better understanding to dew computing and enhance research in this area.

Dew computing applications is the biggest category. "Edge and Dew Computing for Streaming IoT" by Gusev involved IoT streaming applications; "Enhancing Usability of Cloud Storage Clients with Dew Computing" by Mane, Agrawal and Gill proposed an improvement to existing Storage in Dew (STiD) applications; "Vehicular Data Analytics Dew Computing" by Thulasiraman, Thulasiram and Liu used dew computing in intelligent transportation systems; "Dewblock: A Blockchain System Based on Dew Computing" by Wang applied dew computing to blockchain technology and provided source code of such a new application. All these new applications show that dew computing has inspiring power in solving real-world problems.

We want to express our gratitude to our sponsors: IBM Centre for Advanced Studies, CAS-

CON 2018, and the School of Mathematical and Computational Sciences at the University of Prince Edward Island, Canada. Without their generous support, it would not be possible for DEWCOM 2018 to reach its goals. We also want to express our appreciation to Easychair.org; their support makes our program organization much easier.

October 30, 2018
Toronto, Ontario, Canada

DEWCOM 2018 Co-Chairs:
Yingwei Wang, University of Prince Edward Island, Canada
Karolj Skala, Ruder Boskovic Institute, Croatia

Program Committee

Yi Pan	Georgia State University, USA (Chair)
Marjan Gushev	Ss. Cyril and Methodius University, Macedonia
Shuhui Yang	Purdue University Northwest, USA
Ralph Deters	University of Saskatchewan, Canada
Andy Rindos	IBM Emerging Technology Institute, USA
Enis Afgan	Johns Hopkins University, USA
Sven Groppe	University of Lbeck, Germany
Chrysanne Dimarco	University of Waterloo, Canada
Dana Petcu	West University of Timisoara, Romania
Parimala Thulasiram	University of Manitoba, Canada
Karolj Skala	Ruder Boskovic Institute, Croatia
Yingwei Wang	University of Prince Edward Island, Canada

Program Schedule

Session 1	3:15 - 5:15pm. Oct. 29, 2018. Meeting Room Elm 2
Session Chair	Karolj Skala
Tutorial	Dew Computing Introduction (Yingwei Wang and Karolj Skala)
Paper Presentation	Dewblock: A Blockchain System Based on Dew Computing (Yingwei Wang)
Session 2	8:30 - 10:00am. Oct. 30, 2018. Meeting Room Elm 1
Session Chair	Marjan Gusev
Paper Presentation	The Rainbow Global Service Ecosystem (Karolj Skala and Zorislav Sojat)
Paper Presentation	Vehicular Data Analytics Dew Computing (Parimala Thulasiraman, Ruppa Thulasiram and Ying Ying Liu)
Session 3	10:30am - 12:00pm. Oct. 30, 2018. Meeting Room Elm 1
Session Chair	Parimala Thulasiraman
Paper Presentation	Formal Description of Dew Computing (Marjan Gusev and Yingwei Wang)
Paper Presentation	Enhancing Usability of Cloud Storage Clients with Dew Computing (Tushar Mane, Himanshu Agrawal and Gurmeet Sigh Gill)
Paper Presentation	Edge and Dew Computing for Streaming IoT (Marjan Gusev)
Session 4	1:00 - 2:00pm. Oct. 30, 2018. Meeting Room Elm 1
Session Chair	Ralph Deters
Paper Presentation	Overview of Cloudlet, Fog Computing, Edge Computing, and Dew Computing (Yi Pan, Parimala Thulasiraman and Yingwei Wang)
Session 5	2:00 - 3:00pm. Oct. 30, 2018. Meeting Room Elm 1
Session Chair	Yi Pan
Discussion	Dew Computing Development Strategies
Session 6	3:30 - 5:30pm. Oct. 30, 2018. Meeting Room Elm 1
Committee Meeting	DewCom STC Committee 1st Face-to-face Meeting

Edge and Dew Computing for Streaming IoT

Marjan Gusev
University Ss Cyril and Methodius
Skopje, Macedonia
Email: marjan.gushev@finki.ukim.mk

***Abstract*—The main question in the case of streaming data coming from end-user IoT devices with big quantities is where to process data. The available choices are whether the processing of the required information should take place on a local device or to offload data to a nearby or remote server for further processing. Basic IoT schemes include only local processing, while more sophisticated schemes include offloading to nearby servers on the edge of the network, or to remote distant cloud servers. In this paper, we analyze the implementation details and organizational approaches related to dew computing, where the processing is brought even closer to the user than the edge computing concept. The relevant features will be compared to classical edge approaches, such as cloudlets, fog computing, mobile edge computing or similar computer architecture approaches.**

***Index Terms*—Mobile Cloud Computing, Cloudlet, Edge computing, Fog computing, computation offload**

I. Introduction

Computing devices are embedded in almost all end-user devices used in everyday activities. With the growth of the Internet and the advances of modern technology, users can control and use these devices over the Internet. This is the basis of the Internet of Things (IoT) as an organized interconnection of all these devices [1], as independent computing devices that function in a shared environment over the Internet.

In this research, we address issues that arise with IoT devices that generate data with high volumes and velocity, characterizing them in the Big Data concept as they need computing units that provide fast processing and massive storage capacities. In addition, another problem arises when the user tries to use them as independent battery-operated mobile devices with a wireless connection. It requires special designs, so the IoT devices will be relieved of all tasks that consume a lot of energy.

An example of such a device is a wearable eHealth or ECG sensor. It is a small device that can be patched on a user's chest. To make it more comfortable, it needs a very small weight and small size, such that will not cause any obstacles for user daily activities and movements. Therefore, the designers of such a device face the constraint of using a very small battery that should be recharged on a couple of days, for example, a week. The device needs to process a lot of data generated as a 2-byte integer samples on a regular sampling frequency higher than 250 Hz, which will generate a data stream with a rate of 30 KB per minute, and storage demands of 1.8 MB per hour or 54 MB per day. Essential data processing and diagnosis may require up to 500 executable commands per sample, so the processing needs a processing power of at least 125.000 operations per second, excluding the operations required by the operating system. Although it may not look so demanding for a modern computer, still it will spend a lot of energy on a smaller embedded sensor, and will not fit in the constraint for a small battery.

Offloading is a promising alternative, but still, the users are concerned when to offload and where to offload. This paper analyzes several different approaches and architectural designs. A comprehensive comparison is provided to discuss all relevant issues and help a solution provider how to organize the computing, storage in order to minimize energy consumption of the end-user IoT device without degrading the performances of the application.

The main concepts of dew and edge computing are compared to distinguish between different designs and approaches. In this paper, we will explain what is the difference between edge and dew computing, and answer where and when to offload. We will present differences in design and implementation, addressing the application domains.

The paper organization is as follows. Section II gives a state-of-the-art and related work on edge and dew computing architectural concepts. Our view and distinction between different architectural approaches are explained in Section III followed by a discussion in Section IV. Finally, relevant conclusions and future work directions are given in Section V.

II. Related Work

Streaming IoT solutions belong to a wider class of ubiquitous and pervasive computing solutions for IoT devices [2]. A streaming IoT device is considered to be a device that generates at least 100 samples per seconds [3], [4].

The first idea to offload data and computations initiates a cloud server connection to a mobile device. The mobile device is considered to be the end-user IoT device and the cloud server is the computing unit that will process streaming data.

Dinh et al. [5] discuss the advantages of dynamic provisioning, scalability, multitenancy, and ease of integration for related mobile cloud computing applications. Issues that need to be addressed in the mobile cloud computing include low bandwidth, availability, heterogeneity, static and dynamic environments in computation offloading, security, privacy and other quality of service and related open issues.

In addition, the presented architecture does not address wearable mobile IoT devices with limited power supply capabilities and small computing capacities. This is why the edge computing is introduced as an architecture solution [6].

A. Edge Computing

The focus of architectures and computer implementations has shifted towards gaining real-time responses along with support for context-awareness and mobility in the IoT [7], enabled by edge computing.

The edge computing technology promises to deliver highly responsive cloud services for mobile computing, scalability and privacy-policy enforcement for the IoT, and the ability to mask transient cloud outages. Satyanarayanan [8] elaborates that the idea of caching is used in edge computing for caching the cloud services.

Edge computing pushes the cloud services closer to the user and also pulls the IoT micro-services from IoT devices [9]. It changes the vision of data consumer to data producer of an IoT device.

Two approaches dominate the use of edge computing architectural organizations. They differ by the implementation provider [10], [4], so if the mobile operator is providing an infrastructure, then it is a basis of fog computing and if an Internet provider uses LAN networking for the edge devices, then it is a cloudlet solution. Some authors find these terms to be synonyms to edge computing [11].

Satyanarayanan [12] specifies *a cloudlet* as an infrastructure based on a virtual machine located in the proximity of the end-user device accessed in a LAN environment. Verbelen et al. [13] describe that the cloudlets do not have to be fixed infrastructure close to the wireless access point, but can be formed in a dynamic way with any device in the LAN network with available resources.

Cloudlet challenges have been analyzed [14] for their architectural and implementation issues. The corresponding definition clearly identifies another architectural layer between the cloud server and the end-user device.

Bonomi et al. [15] define essential *fog computing* concepts setting servers at the base stations to reduce the latencies and distribute the processing in a number of IoT applications.

Chiang and Zhang [16] analyze the latency requirements and bandwidth constraints in the context of IoT resource-constrained devices, along with other non-functional issues including security and protection.

Some authors use the fog computing concept simply to present the realization of computer communication infrastructure with routers, switches, access points, and gateways, and others as computing nodes at the edge of the mobile network. Therefore, some authors think that the fog concept is equal to the edge computing concept in the context of computing, similar to the concept of servers used in mobile edge computing. Other authors observe the fog computing only as a communication infrastructure.

Another related concept is the *mobile edge computing* aiming at reducing network stress by shifting computational efforts from the Internet to the mobile operator's edge. Although according to the previous understanding of fog computing as an edge computing communication infrastructure, the complete idea of mobile edge computing is an application of the fog computing concept.

An early definition of mobile edge computing can be found in several papers, although some of these definitions in our context are a specification of dew computing. For example, Kim et al. [17] introduce the concept of Mobile Edge Computing Devices as an interface between distributed sensors and the end server in order to reduce processing and bandwidth requirements to the end servers, and provide enhanced scalability, flexibility, reliability, and cost-efficiency.

ETSI [18] tries to standardize it as a key technology towards 5G, to provide an IT service environment and cloud-computing capabilities at the edge of the mobile network and in close proximity to mobile subscribers, aiming at reducing the latency and improving the user experience. In addition, ETSI [18] designs it as a natural development in the evolution of mobile base stations and the convergence of IT and telecommunications networking by using virtualized environments.

Mobile edge computing is able to provide IoT services, which are not technically or economically feasible otherwise. In addition, bringing mobility support functions to the mobile edge platform may have a dramatic impact on the existing architecture.

Mobile edge computing architecture has been analyzed by Beck et al. [19] and a taxonomy is specified according to the following criteria: offloading, local connectivity, content scaling, augmentation, edge content delivery, and aggregation.

Particularly, according to their definition, cloudlets, can also use offloading to a mobile edge computing server, which in our case is a definition of a dew computing layer. The difference in specifying it as a non-cloudlet layer lies in our definition that a cloudlet server is a representation of an edge computing layer, and it is on the same level to the mobile edge server.

Wang et al. [20] conclude that mobile edge networks provide cloud computing and caching capabilities at the edge of mobile operator networks.

B. Dew Computing

Dew computing has been defined by several research papers [21], [22], [23] as an architecture that brings computing closer to the user. Wang et al. [24] discuss the transition of Internet computing paradigms towards dew computing.

Dew computing concepts are complementary to the edge computing concept. We define dew computing concept for streaming IoT devices as end-user devices that do not have Internet access via LAN network in order to transfer data or offload computations to an edge or cloud server.

The basic definition of dew computing concept [22] specifies two essential features:

- *independence*, by enabling an environment where the IoT device can perform locally and interact with the end-user without the need of a permanent Internet connection, and
- *collaboration*, by enabling an environment where the IoT device can collaborate with other devices via an Internet connection.

Ray [25] discusses that the independence and collaboration features in the Wang definition [22] need an addition of the microservice concept [21] although it is indirectly assumed

that a microservice is the key feature to allow independence feature. The independence concept and new formulation of the collaboration feature have been analyzed by Ristov et al. [23], where the information-centric feature is added to the existing two essential features, although, they can be treated as an indirect.

A specification of a dew computing architecture was given by Wang [26] along with an elaboration of functional requirements. It is an extension of a cloud-based client-server architectural concept adapted to a new environment.

A dew server is defined by Wang [26] as a tiny light-weight server that provides microservices [21]. Ray [25] enhances it with a more detailed specification. He discusses three types of novel-services: infrastructure-as-a-dew, software-as-a-dew service, and software-as-a-dew product. In addition, he defines that the dew computing model is composed of six essential characteristics: Rule-based Data Collection, Synchronization, Scalability, Re-origination, Transparency, and Any Time Any How Accessibility.

In this paper we address dew computing application in IoT, especially targeting the streaming IoT devices. An overview of dew computing solution for streaming IoT is presented in [3]. The implementation details address the way the devices connect to each other in various environments. The communication can be established directly from the IoT devices to the cloud server, or to the edge devices via various personal area network or LAN technologies.

III. Architectural Concepts

We start with a basic client-server architecture, where a client (a computing device located in a lower architectural layer) is connected to a server (located in the upper architectural layer) by a communication link. The edge computing concept introduces an intermediate server, called edge server between the cloud server and the client, which in our case is a streaming IoT device.

According to the provider of the communication infrastructure, there are two approaches of the edge computing architecture, the first based on a cloudlet edge server and the second on edge servers mobile operator's network.

A. Cloudlet Edge Computing Architecture

A simple design of adding an edge server between the end-user IoT device (client) and the cloud server is based on a cloudlet. A cloudlet is a smaller server on the edge of Internet network provided by an Internet provider. The cloudlet server collaborates with the end-user IoT device by a LAN technology. Its main function is to provide services to the client, since the end-user streaming IoT device can neither perform complex computations nor store big amounts of data. In addition, it may be a mobile device which is wirelessly connected to the edge server and is battery-operated, so its function is constrained by the capacity of an installed battery.

The cloudlet edge computing concept is presented in Fig. 1. The top layer consists of a cloud server, the lower layer of IoT devices. The intermediate layer specifies the cloudlet edge

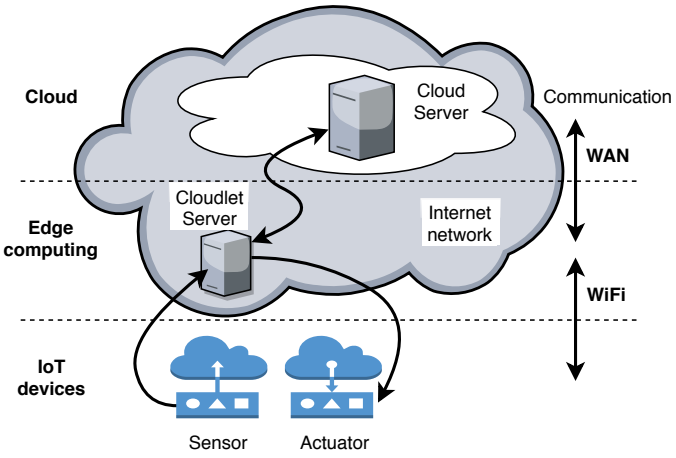


Fig. 1. Cloudlet edge computing architectural approach

server. The communication between the cloudlet and cloud server is based on WAN and between the cloudlet and IoT devices on WiFi LAN networking.

The IoT device offloads data and computations to the nearby cloudlet. It relieves the energy supply demands since complex computations and data storage are transferred to the cloudlet. Since the communication is local via a wireless network, the expected delays are relatively small and are much lower than standard WAN delays used in the case of a remote cloud server.

B. Mobile Edge Computing Architecture

The second approach of building an edge computing solution is based on using an edge server on the mobile operator's network. When analyzed from an architectural design view, this is the same three-layer architecture, where an edge server is added to the client-server architecture. The difference to the cloudlet approach is that the communication infrastructure is provided by the mobile operator. The edge server is located on the edge of the Internet perimeter of the mobile operator, while the IoT devices are using the mobile operator's network.

The presented design differs from the previous since the edge server is not anymore a cloudlet, it is another server owned by the mobile operator and located at the base station on the edge of Internet network. The whole communication between mobile operator's cloud server and the edge server at the base station is via WAN provided by the mobile operator. The end-user IoT device communicates with the edge server via radio waves of the mobile operator's network, such as 3G/4G or 5G.

Fig. 2 presents a mobile edge computing architectural approach. The IoT device can offload data and computations to a more powerful edge server. Since it is located at the base station of the mobile operator's network, the expected transmission delay is very small and the IoT device performance is relatively high.

The problems associated with streaming IoT and enabling their mobility by wireless connection and small size battery-operated function cannot be solved by classical edge com-

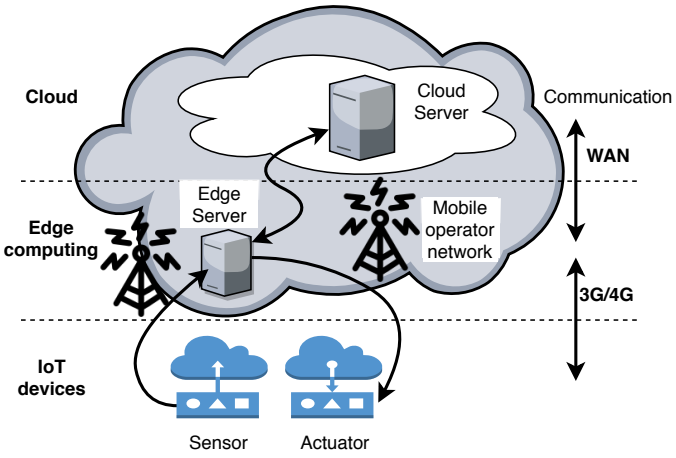


Fig. 2. Mobile edge computing architectural approach

puting solutions. It cannot be solved either using cloudlets nor direct mobile edge computing architecture. The solution is provided by adding a new layer between the edge server and the IoT device. Similar to the previous categorization, two approaches are possible, the first by using a LAN connected cloudlet or an edge server on a mobile operator's network.

C. Dew Computing Cloudlet Architecture

Adding another dew computing layer in the cloudlet architecture is presented in Fig. 3. A streaming IoT device with limited power supply and mobility enabled with personal area networking wireless connectivity can communicate only with a dew server. A dew server is out of Internet perimeter outside the edge of a network and therefore it cannot be treated as an edge device. It communicates to the cloudlet edge server via LAN. The cloudlet server can also communicate with other cloudlet or cloud servers to exchange results and any relevant information.

The IoT device streams data to a nearby dew server via Bluetooth or other personal area network communication. It neither performs any computation nor stores any data. It is a simple realization of a sensor that senses a signal and transfers data to the nearby dew server. The dew server takes the role of essential signal processing and data storing. In addition, it is able to transmit data and complex computations to a cloudlet server, or exchange information about the outside world.

In the context of dew computing, the dew server performs its functions independently and can collaborate with other devices.

D. Dew Mobile Edge Computing Architecture

The dew mobile edge computing architectural design differs from the cloudlet dew computing architecture in the communication of the dew server with the edge server. Instead of a cloudlet on the LAN provided by the Internet provider, the design uses an edge server on the edge of the mobile operator's network. A mobile operator's radio communication is used for communication between the edge server and dew server,

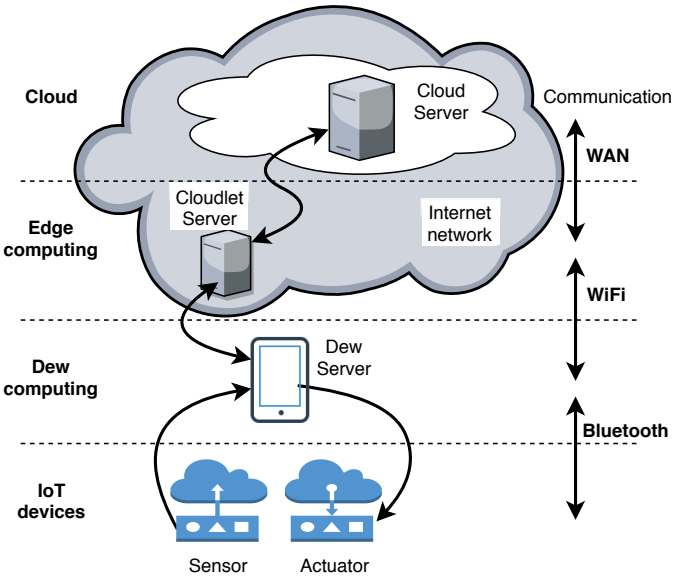


Fig. 3. Cloudlet dew computing architecture

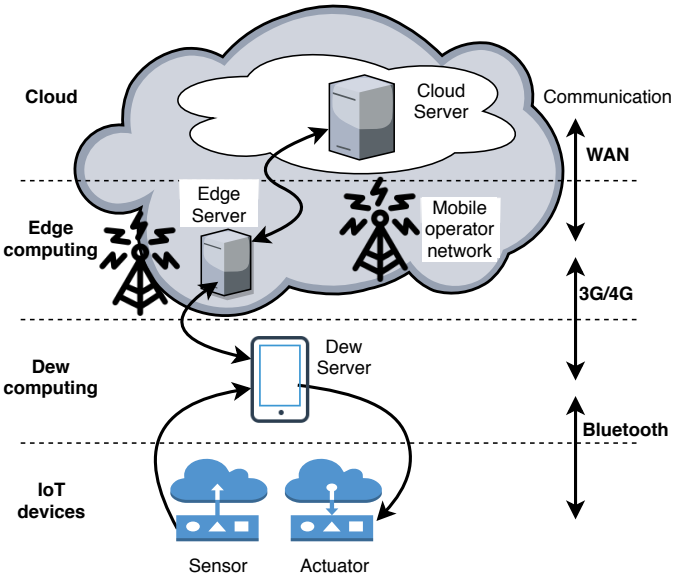


Fig. 4. Dew mobile edge computing architecture

instead of LAN used as a communication between the cloudlet edge server and dew server.

Fig. 4 presents the dew mobile edge computing architecture. The IoT device is a light mobile device wirelessly connected with limited battery-operated power supply. It can communicate with a dew server via Bluetooth or any other personal area network communication link. The dew server uses 3G/4G or 5G radio communication link established by the mobile operator. The edge server found on the edge of the mobile operator's Internet network can communicate with the main cloud server to exchange information.

Some readers may argue that this is another edge computing

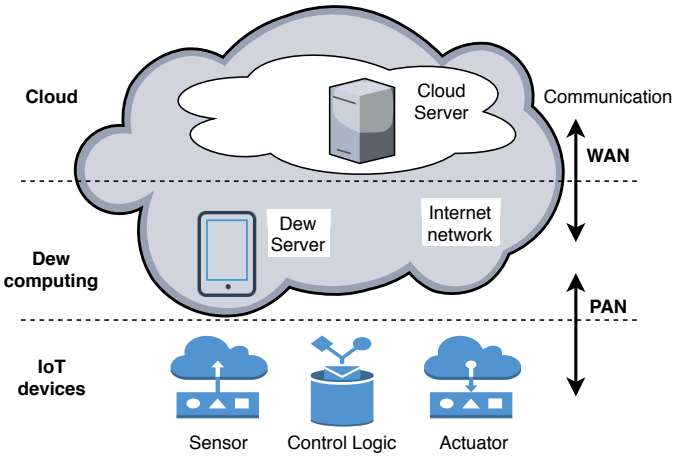


Fig. 5. A reduced dew computing architecture

implementation. Let's specify the main differences. The dew server is out of the Internet network, it uses 3G/4G or 5G to communicate to the edge server and therefore is out of the Internet edge, although, indirectly it is connected to the Internet. In addition, the IoT device uses a personal area network to connect to the dew server, instead of the mobile operator's radio network.

The analyzed issues make a clear distinction between the two identified edge computing approaches and these two dew computing approaches. Using another dew computing layer between the edge server and the IoT device results in enabling an environment for light mobile streaming IoT devices that spend only a small portion of energy for a low power personal area network communication link.

E. Dew Computing Cloud Architecture

To be consistent we will provide another dew computing architecture which does not belong to the previous edge computing architectures (cloudlet and mobile edge computing). It is based on direct communication between the dew server and the cloud, as presented in Fig. 5. A streaming IoT device is a light mobile IoT device wirelessly connected to the dew server, which is capable to establish a WAN connection to the cloud server.

There is a great similarity between the direct dew cloud architectural approach and the cloudlet edge computing architecture. The difference is in the communication between the IoT device and the dew or edge server. A dew server is capable to accept personal area network communication, while the edge device only WiFi LAN network. Personal area communications are required by the streaming IoT device to save the energy consumption. Since it is a small mobile device and it is intended to spend only a small portion of energy, it cannot support WiFi connection, but only low energy Bluetooth or similar local radio connection.

It also explains the similarity to the mobile edge computing architecture, since it is using 3G/4G or 5G mobile operator's communication link instead of low energy Bluetooth or similar local radio connection.

IV. Discussion

Different analyzed architectural approaches are compared to implement an effective solution for streaming IoT devices that demand mobility, low power local wireless communication and battery-operated devices that spend only small energy for its performance.

A. Edge vs Dew computing approaches

According to Wang [26], a dew server is added on a path between the client and cloud server. In addition, it can work independently and collaborate with others.

However, the idea of adding a server between the client and the cloud server fits more to the edge computing concept. By definition, edge computing brings the computing to the edge of the network, so a new edge server is located next to the client, that is the streaming IoT device in our case.

Comparing these two approaches, dew computing is an extension of the edge computing concept, not the client-server concept, as elaborated in the previous Section.

The evolution of edge computing concepts and their implementation in IoT has been discussed by Gusev and Dustdar [4]. Two approaches are defined in the case processing is realized on the edge of the network, provided either on the mobile operator's network or on the LAN provided by an Internet provider. The dew concept is defined by bringing the processing even closer to the IoT device than the edge computing concept. The end-user IoT device is not on the edge of the network but will communicate to an edge device that will provide a connection to the Internet and all relevant functionalities.

Zhou et al. [27] define post-cloud computing paradigms to include fog computing, mobile edge computing, and dew computing. According to their definition, fog computing is a horizontal architecture for a virtualized platform that provides computation, storage, and services between end devices and cloud servers, which slightly differs to our understanding that fog computing refers only to a communication infrastructure environment. Further on they define mobile edge computing as an architecture offered at the edge of a mobile network. Although these two items differ in their definition, we can conclude that they have specified fog computing as a specialized virtualized environment and mobile edge computing as an environment provided on the edge of the mobile operator's network. In our definition, both are considered as part of edge computing, and fog is a synonym to edge, or its special case when virtualized environments are used.

As a special form of a post-cloud computing paradigm [27], dew computing is specified as a software organization model where local computers provide rich functionality independent of cloud services. So this fits into our definition that they are not on the edge of the network, and these devices can communicate to edge devices and access edge or cloud servers to exchange information or even offload. data and computing.

Another interesting feature is the communication access to end and intermediate devices. Some authors [7] define fog computing as an infrastructure where all communication mechanisms are supported, including PAN (Bluetooth), LAN (WiFi) and mobile operator networks (3G/4G). Although most of them set an equivalency between the fog computing and edge computing concepts, the main difference to the mobile edge computing as they only use mobile networks (3G/4G), while cloudlet implementations use only LAN (WiFi) connection. We do not agree to this classification and define the dew computing layer, which communicates to the IoT layer by PAN (Bluetooth), and to the above layer via LAN (WiFi) in the cloudlet implementation or via mobile operator network (3G/4G) in the mobile edge computing concept.

According to the classification presented by Dolui and Datta [7], the context awareness in fog computing is medium, and in mobile-edge computing high, while the cloudlet context awareness is low. We do not agree to this classification and add that a dew-computing cloudlet solution may also have high context awareness.

Zhou et al. [27] discuss that edge computing is just another computing paradigm advocated by the academic community [28] with a wider description and broader meaning than fog computing. Actually, edge computing is based on fog computing and refers to and is included in the definition and categories of mobile edge computing, as a general term that covers both fog computing and mobile edge computing. However, there is no broad consensus on the concept of edge computing.

B. Dew computing challenges for IoT streaming devices

According to several other definitions of dew computing, the dew devices does not have permanent LAN connection and Internet access, but can have occasionally. However, in our specification of dew computing solution for streaming IoT devices, these devices can collaborate with edge devices via personal area network and, therefore, access the wider world via LAN network and Internet connectivity.

Analyzing the basic definition of dew computing devices with features to perform independently and collaborate with other devices, the edge computing concept provides only the second feature, since it must have an Internet or other connection to the server where the processing of offloaded computation is performed and where the massively collected data is stored. However, in our case, a dew device is a streaming IoT device without direct connection to the server and can perform regularly without communicating a remote edge or cloud server. Connecting to an edge device via personal area network technologies will enable indirect Internet availability.

Note that although a streaming IoT device can process data locally, still it may not perform all required functions that need an exchange of information with the wider world. For example, if the streaming IoT device is not connected to an edge device, it may lose its data and fail to deliver results to the outside world.

Difference between the classical implementation of a mobile-edge computing solution and its dew computing implementation is in the way the IoT devices are used. The conventional approach means that the smartphone or any other mobile device is at the same time an end-user device and IoT device. However, the dew computing implementation introduces two layers, instead of one, the first is the IoT devices layer, and the second is the dew computing layer with smartphones or other mobile devices.

Analyzing the edge and dew computing architecture, we can conclude that they change the centralized approaches to distributed decentralized environment. In addition, Nastic et al. [29] define a serverless real-time data analytics platform where the (micro) services provided by edge devices, edge servers or, in our case, dew servers, can be transferred (offloaded) to another device found in the nearby proximity and ready to accept its function.

The main challenges in dew computing architectures that support streaming IoT devices are:

- autonomous functioning, by communication to a local dew server
- lower latency and high bandwidth communication, and
- minimal energy consumption, by minimizing the number of operations, data storage and data transfer.

V. Conclusion

To bring the processing closer to the end-user streaming IoT devices we can use the concept of edge computing. The edge in this context has LAN connectivity, and Internet access provided either by a mobile operator or by an Internet provider.

We have analyzed a typical case when end-user streaming IoT devices are constrained by mobility, wireless connectivity, and limited battery-operated power supply. These features prevent using the edge concept directly, so another architectural concept needs to be designed to cope with these issues. The presented solution belongs to the dew computing concept.

Dew devices are the end-user streaming IoT devices that can perform required activities independently and collaborate with neighboring edge devices to access a LAN network and Internet connection. We have presented architectural design details on these dew concept implementations in both cases where the edge is provided by a mobile operator or Internet provider.

The main difference between the edge and dew computing concepts are in the definition of the location of the end-user device. If it is on the edge of the Internet network, then it is treated as an edge computing architecture, while if it is outside of the Internet perimeter edge, then it is a dew computing architecture. Both architectural styles aim at bringing the computing closer to the user, and we can say that dew computing brings it even closer to the user. This fits in the definition of going back to the roots that are to the end-user devices.

In the case the end-user device demands mobility, low energy local radio wireless connection performing as small processing tasks as it is possible to save the battery-operated

power supply, then the answer is in the use of a dew computing architecture. The dew server design can be also constrained by low energy consumption to save the energy and may use LAN or mobile operator radio communication links, determining if the solution will use cloudlet or mobile edge networking.

REFERENCES

[1] O. Vermesan, P. Friess, P. Guillemin, S. Gusmeroli, H. Sundmaeker, A. Bassi, I. S. Jubert, M. Mazura, M. Harrison, M. Eisenhauer *et al.*, "Internet of things strategic research roadmap," *Internet of Things-Global Technological and Societal Trends*, vol. 1, no. 2011, pp. 9–52, 2011.

[2] M. Satyanarayanan, "Pervasive computing: Vision and challenges," *IEEE Personal communications*, vol. 8, no. 4, pp. 10–17, 2001.

[3] M. Gusev, "A dew computing solution for IoT streaming devices," in *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2017 40th International Convention on*. IEEE, 2017, pp. 387–392.

[4] M. Gusev and S. Dustdar, "Going back to the roots: The evolution of edge computing, an IoT perspective," *IEEE Internet Computing*, vol. 22, no. 2, pp. 5–15, 2018.

[5] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless communications and mobile computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[6] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future generation computer systems*, vol. 29, no. 1, pp. 84–106, 2013.

[7] K. Dolui and S. K. Datta, "Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing," in *Global Internet of Things Summit (GIoTS), 2017*. IEEE, 2017, pp. 1–6.

[8] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[9] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[10] I. Stojmenovic and S. Wen, "The fog computing paradigm: Scenarios and security issues," in *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*. IEEE, 2014, pp. 1–8.

[11] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[12] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE pervasive Computing*, vol. 8, no. 4, 2009.

[13] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Cloudlets: Bringing the cloud to the mobile user," in *Proceedings of the third ACM workshop on Mobile cloud computing and services*. ACM, 2012, pp. 29–36.

[14] A. Bahtovski and M. Gusev, "Cloudlet challenges," *Proceedia Engineering*, vol. 69, pp. 704–711, 2014.

[15] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. ACM, 2012, pp. 13–16.

[16] M. Chiang and T. Zhang, "Fog and iot: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016.

[17] S. J. Kim, G. Deng, S. K. Gupta, and M. Murphy-Hoye, "Enhancing cargo container security during transportation: A mesh networking based approach," in *Technologies for Homeland Security, 2008 IEEE Conference on*. IEEE, 2008, pp. 90–95.

[18] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computinga key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015.

[19] M. T. Beck, M. Werner, S. Feld, and S. Schimper, "Mobile edge computing: A taxonomy," in *Proc. of the Sixth International Conference on Advances in Future Internet*. Citeseer, 2014, pp. 48–55.

[20] 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*, vol. 5, pp. 6757–6779, 2017.

[21] K. Skala, D. Davidovic, E. Afgan, I. Sovic, and Z. Sojat, "Scalable distributed computing hierarchy: Cloud, fog and dew computing," *Open Journal of Cloud Computing (OJCC)*, vol. 2, no. 1, pp. 16–24, 2015.

[22] Y. Wang, "Definition and categorization of dew computing," *Open Journal of Cloud Computing (OJCC)*, vol. 3, no. 1, pp. 1–7, 2016.

[23] S. Ristov, K. Cvetkov, and M. Gusev, "Implementation of a horizontal scalable balancer for dew computing services," *Scalable Computing: Practice and Experience*, vol. 17, no. 2, pp. 79–90, 2016.

[24] Y. Wang, K. Skala, A. Rindos, M. Gusev, S. Yang, and Y. PAN, "Dew computing and transition of Internet computing paradigms," *ZTE COMMUNICATIONS*, vol. 15, no. 4, 2017.

[25] P. P. Ray, "An introduction to dew computing: Definition, concept and implications," *IEEE Access*, vol. 6, pp. 723–737, 2018.

[26] Y. Wang, "Cloud-dew architecture," *International Journal of Cloud Computing*, vol. 4, no. 3, pp. 199–210, 2015.

[27] Y. Zhou, D. Zhang, and N. Xiong, "Post-cloud computing paradigms: a survey and comparison," *Tsinghua Science and Technology*, vol. 22, no. 6, pp. 714–732, 2017.

[28] Open Edge Computing, "Open edge computing initiative," 2017. [Online]. Available: http://openedgecomputing.org/index.html

[29] S. Nastic, T. Rausch, O. Scekic, S. Dustdar, M. Gusev, B. Koteska, M. Kostoska, B. Jakimovski, S. Ristov, and R. Prodan, "A serverless real-time data analytics platform for edge computing," *IEEE Internet Computing*, vol. 21, no. 4, pp. 64–71, 2017.

Formal Description of Dew Computing

Marjan Gusev
Ss. Cyril and Methodius University,
Faculty of Information Sciences and Computer Engineering,
Skopje, Macedonia
Email: marjan.gushev@finki.ukim.mk

Yingwei Wang
University of Prince Edward Island,
School of Mathematical and Computational Sciences
Charlottetown, Canada
Email: ywang@upei.ca

***Abstract*—Dew Computing is a specific cloud-related computing architecture that brings the computing closer to the user. Two main features of the dew computing include independence of external systems and collaboration with other cloud servers, making it an environment that can work in two modes, localized mode where all the services are provided within the internal local network perimeter and global mode, where it functions just as an intermediate device in the client-server cloud model. This article presents a formal description of dew computing as a service model and defines its two main operating modes with mathematical modeling functions.**

***Index Terms*—Dew computing; Cloud computing; Formal specifications; Service modeling; Computational modeling; Turing machines; Servers.**

I. Introduction

Dew computing is an on-premises computer software-hardware organization paradigm in the cloud computing environment where the on-premises computer provides functionality that is independent of cloud services and, in the same time, it is collaborative with cloud services, too.

The goal of dew computing is to fully realize the potentials of on-premises computers and cloud services. Here, an on-premises computer is a cloud computing term. It means local computers, or non-cloud computers, which include personal computers (desktops, laptops), tablets, smartphones, servers, and clusters.

Wang [1] explains the essential dew architecture to be an extension of a classical "client-server" architecture concept by introducing an intermediate dew server located close to the client. Although this looks similar to the cloudlet concept [2] as an edge computing concept [3], we will discuss the distinctions among them in Section VI.

Dew computing has two major features: *independence* and *collaboration* [4]. Independence means the on-premises computer is able to provide functionality offline. Collaboration means the dew computing application has to automatically exchange information with cloud services during its operation. Such collaboration includes synchronization, correlation, or other kinds of inter-operation.

Ray [5] discusses that besides these two features, two more features characterize dew computing. The first one addresses the "microservice provision", as defined by K. Skala et al.[6], which incorporate the theory of micro-service components located far away from ent virtual infrastructures. The second one specifies the "scalability" in correlation to "independence" and "collaboration" as analyzed by Ristov et al. [7].

Dew computing is an emerging research area and application area. Although the theory and methods of dew computing are being shaped, many dew computing applications have already existed for many years, even before the dew computing concept was proposed.

To clarify the concept of dew computing and to further facilitate dew computing applications, we need to precisely determine which applications are dew computing applications. In other words, we need a model to describe dew computing applications. In this paper, we try to develop such a formal specification and computing model.

The rest of the paper is organized as follows. Section II gives the background and describes the dew computing modeling considerations. A model specification of a generic server model is presented in Section III and formal definitions of a dew service and dew server systems are introduced in Section IV. Examples of simplified modeling using service resources are elaborated in Section V. Section VI discusses the derived model and compares our formal specification with related approaches. Finally, Section VII is devoted to conclusions and future work correspondingly.

II. Background

This section elaborates a background for developing a formal specification of dew computing, including dew computing modelling considerations and service model essentials.

A. Dew Computing Modeling Considerations

A dew computing model should satisfy several requirements. For example, at least it should cover all forms of dew computing applications provided as a service system, and should not be restricted by a special group of applications.

For example, one simple way to model dew computing is that every dew computing system is considered an *internet*. Here the word internet starting with a lowercase *i* indicates that this is a group of computers that are connected through TCP/IP protocols. Considering this approach, each on-premises computer or a group of such computers are organized as an internet; websites are created on this internet; various services exist in this independent dew world.

In this model, all the dew applications communicate with the cloud which is the *Internet* with uppercase *I*; the relationship

between a dew and the cloud is actually the relationship between a small internet and the big Internet. Although an internet and the Internet are different in their sizes, they are equal in terms of structure: they are both governed by TCP/IP protocols; the collaboration feature between a dew and the cloud can be interpreted as the communication among different internets.

This model is very practical and useful. It covers a broad range of applications, including dew servers and dewsites of the Web in Dew (WiD) applications.

This model has its drawbacks. The biggest problem is that It has limitations; only those applications that conform to TCP/IP protocols are covered. Theoretically, dew computing can be implemented using techniques other than those using TCP/IP protocols.

Analyzing the application domains, Rindos and Wang [8] identify Web in Dew (WiD) and Infrastructure as Dew (IaD) categories of dew computing. Later on, new categories are identified as Storage in Dew (STiD), Database in Dew (DiD), Software in Dew (SiD), Platform in Dew (PiD), and Data in Dew (DaD). Also, several research papers include dew computing in IoT architectures and applications [9].

Based on the above discussions, we need a dew computing model that covers a broader range of dew computing applications, that does not directly involve technical implementation details.

B. Server and service provision model

We consider that a service is provided by a complex system that interacts with other systems and performs a transformation of the requests to provide the output. This complex system is considered as a service provision system, and generally, is called a server system. Fig. 1 presents a basic model of the server that provides a generic service.

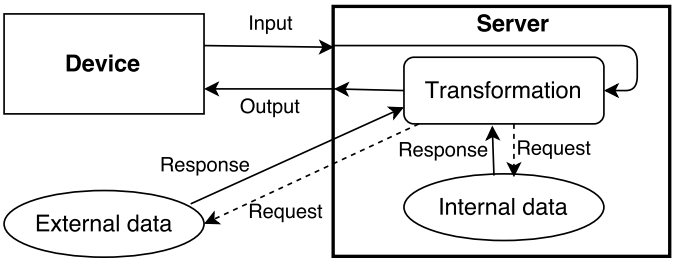


Fig. 1. Model of a basic server system providing a service to the device using external systems to compute results

To understand the basic model, Fig. 1 also contains a description of a device, which can be any computing or another device that can generate a service request (defined as an input to the server) and obtain a service response (defined as an output of the server). In a classic client-server model, the client is any computing device that sends a service request, and the server is the computing unit that generates a service response.

In addition, Fig. 1 specifies an external system which in essence is also another server system that generates a service response to a given service request. However, in this case, the analyzed server generates a service request as an input to the external system and expects a service response generated by the output of the external service system. The same terms that are associated with input and output to the external server system, in this case are treated as output and input correspondingly to the analyzed server system.

Let's dig deeper into the architecture of the basic server system and its service provision. Internally, the analyzed server system receives a service request defined by an input data I and calculates an output data O that is transferred as service response to the requestor device. The processing of the input uses a set of data transformations. As this is a computing system, it can generate the output based on the input data if the system complies to a definition of a combinatorial logic only.

However, the presented generic model of a server machine is a more complex system, and besides the input, it computes the output, also based on its internal state. In this case, the model of this computing machine uses finite state automata and the concept of memory that stores internal data. The memory itself can be treated as another service system that accepts a service request as an input and outputs a service response by providing data. Once again the communication to the memory is by a service. The analyzed server generates a memory service request to the memory in order to access the internal data and receives a service response as an output of the memory.

The overall server model also uses external data generated by the external server systems, as discussed earlier. This summarizes the definition of a generic server system to calculate an output based on three different inputs:

- *data input* I generated by a client device (service request),
- *internal data* M provided by memory as an internal service, and
- *external data* E provided by external services.

As a conclusion, a generic server system depends on other server systems, such as memory and external service providers. It transforms the data input I, using internal data M and external data E provided by corresponding service providers to generate output data O,

Therefore, a typical modeling will define a relation between the output O and inputs I, M, E by a specific transformation function ω. In addition, it will change internal data in the memory by a specific state transition function δ. Details on the development of a service model are specified in the next section.

III. A Generic Server Model

Our definition of a server model will be given in relation to a specification of a Turing machine.

A. Turing Machine definition

Now, let's analyze the connection of such a simple system with a definition of a Turing machine or other models of computations.

A (one-tape) Turing machine, according to Hopcroft and Ullman [10], can be formally defined as a 7-tuple $M = \{Q, \Gamma, b, \Sigma, \delta, q_0, F\}$ where

- Q is a finite, non-empty set of states,
- Γ is a finite, non-empty set of tape alphabet symbols,
- $b \in \Gamma$ is the blank symbol (the only symbol allowed to occur on the tape infinitely often at any step during the computation),
- $\Sigma \subseteq \Gamma \setminus \{b\}$ is the set of input symbols,
- $\delta : (Q \setminus F) \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is a partial function called the transition function, where L is left shift, R is right shift. (A relatively uncommon variant allows "no shift", say N, as a third element of the latter set.) If δ is not defined by the current state and the current tape symbol, then the machine halts.
- $q_0 \in Q$ is the initial state,
- $F \subseteq Q$ is the set of final or accepting states. The initial tape contents will be accepted by M if it eventually halts in a state from F.

Anything that operates according to these specifications is a Turing machine. Any service we are analyzing in this paper is a Turing machine.

B. Specification of a generic server

Now let's correlate our conceptual model of the device and server systems to a definition of a Turing machine as a model of computation or other models of computers. A conventional Turing machine uses unlimited sequential memory, while real computers use limited random access memory. In computer science, random-access machine (RAM) is an abstract machine model identical to a multiple-register counter machine adding indirect addressing. Its equivalency to the universal Turing machine can be observed if the RAM's program and data are stored in the registers realizing a so-called von Neumann architecture.

We will give a mathematical definition of a server system that will be an approximation of a final state machine definition, that will reflect the modeled system behavior.

Definition 1: A general mathematical model of a *server system* is defined by:

- Θ is a finite non-empty set of data values called the data alphabet,
- Σ is a finite non-empty set of input symbols called the input alphabet, such that $\Sigma \subseteq \Theta$
- Γ is a finite non-empty set of output symbols called the output alphabet, such that $\Gamma \subseteq \Theta$
- S is a finite non-empty set of states,
- λ is a finite set of input data values, such that $\Lambda \subset \Sigma$,
- μ is a finite set of memory data values, such that $\mu \subset \Theta$,
- ν is a finite set of data values obtained by the external server systems, such that $\nu \subset \Theta$,
- ρ is a finite set of output data values obtained as a response of the server, such that $\rho \subset \Gamma$,
- s_0 is the initial or start state, such that $s_0 \in S$
- δ is a transition function given by (1)

$$\delta : S \times \Sigma \times \Theta \times \Theta \to S \tag{1}$$

- ω is an output function given by (2)

$$\omega : S \times \Sigma \times \Theta \times \Theta \to \Gamma \tag{2}$$

Θ is the set of all data values that can be treated as an input, output, and memory data value. In reality, it may represent any string, number or structured data. Σ and Γ are sets that represent the possible input and output values used by the system. The triple (Θ, Σ, Γ) consists of alphabets of all possible data used by the system, correspondingly as memory values, accepted input and generated output. In addition, the set of all possible states is S.

The quadruple (s_0, λ, μ, ν) consists of the initial system state s_0, actual input λ, requested memory values μ, requested external data values ν.

Two functions determine the behavior of the system.

Each server functions as a kind of finite state automaton and its behavior is determined by the state in which the automaton is currently in, usually known as an initial state s_0. The transition function actually defines the final (exit) state s_1 of the finite state automaton by providing a function (3) in a deterministic finite automaton. Instead of generating one possible exit state the transition function may return a set of states $\delta \subseteq S$ instead of one state s_1.

$$s_1 = \delta(s_0, \lambda, \mu, \nu) \tag{3}$$

The output function (4) is calculating the set of output values that the server system responds with.

$$\rho = \omega(s_0, \lambda, \mu, \nu) \tag{4}$$

It can be conventional that both the transition and output functions may not be defined on all possible states and data values defined by the input, memory or obtained by an external server system.

The behavior of the server system as a finite state automaton can be modeled as a Mealy machine, since its output depends on the input and internal state, and if the output depends only on the current state then it can be modeled as Moore machine. One may argue that the output function is obsolete if a proper conversion from a Moore to an output-equivalent Mealy machine (by labeling every edge with a transition symbol). However, we prefer to use this definition since it gives all relevant information for a faster understanding of the server system behavior.

This model of the server system has the same computational power as the Turing machine restricted to perform only read operations and moving in one direction only.

IV. A Dew Server formal specification

A. Dew server modeling

A dew server is a specific server system with certain restrictions on the presented model. The main difference is that the dew server can work in a closed environment without the use of external server systems.

Therefore, a dew server's definition is the same as the previous one, but it needs to be extended with the availability of the external server systems and reflect Internet connectivity.

Let's denote by $\eta \in \mathcal{B} = \{0, 1\}$ the availability of Internet connection and the external server systems, by a simple rule that 0 means no availability, and 1 the availability.

The transition function will be modeled by (5) and the output (exit) state will be given by (6).

$$\delta : S \times \Sigma \times \Theta \times \Theta \times \mathcal{B} \to S \tag{5}$$

$$s_1 = \delta(s_0, \lambda, \mu, \nu, \eta) \tag{6}$$

Note, that in the case of a nondeterministic finite automaton the output may be a set of states Δ, instead of one state s_1.

Similarly, the output function ω will be modeled by (7) and the output data set will be given by (8).

$$\omega : S \times \Sigma \times \Theta \times \Theta \times \mathcal{B} \to \Gamma \tag{7}$$

$$\rho = \omega(s_0, \lambda, \mu, \nu, \eta) \tag{8}$$

If the availability is $\eta = 1$ then the dew server model is exactly the same as the general server model. However, if the availability is $\eta = 0$ then the requested external server system value will be undefined or get a *not available* data value N/A. Therefore, the definition of all possible data values that the dew server is operating will need to be extended to Θ' defined by (9). This also applies to the sets of input and output values, correspondingly Σ' and Γ' as given by (9).

$$\begin{aligned} \Theta' &= \Theta \cup \{N/A\} \\ \Sigma' &= \Sigma \cup \{N/A\} \\ \Gamma' &= \Gamma \cup \{N/A\} \end{aligned} \tag{9}$$

This means that in the case of unavailability of external server systems ($\eta = 0$), the associated value that the system will continue to use is $\nu = N/A$ and correspondingly the output of the dew server will be N/A. So, the dew server still performs the specified functions, but in the case of unavailability, it gets a specific value. However, not all server requests will need a value from external servers, and in this case, the dew server will continue to function as it was initially intended for.

Definition 2: A *dew server system* is a server system with the following constraints to the Def. 1.

- Θ' is a finite non-empty set of data values called the data alphabet, defined by (9),
- Σ' is a finite non-empty set of input symbols called the input alphabet, defined by (9),
- Γ' is a finite non-empty set of output symbols called the output alphabet, defined by (9),
- λ is a finite set of input data values, such that $\Lambda \subset \Sigma'$,
- μ is a finite set of memory data values, such that $\mu \subset \Theta'$,
- ν is a finite set of data values obtained by the external server systems, such that $\nu \subset \Theta'$,
- ρ is a finite set of output data values obtained as a response of the server, such that $\rho \subset \Gamma'$,
- δ is a transition function given by (5)
- ω is an output function given by (7)

Finally, we define dew computing based on dew server systems:

Definition 3:

If dew server systems are used in a computing process, this computing process is called *dew computing*.

V. Examples of simplified modeling of service resources

In this section, we will present examples of simplified models of the service resources. A service resource is either an internal memory or external server.

A memory is an internal resource, realized as a set of memory locations. A small part of the memory is used in conventional computers as internal registers, which is a smaller memory located next to the processing unit. In this sense, a memory contains a larger number of memory locations. Each memory location can be accessed by specifying its address and specifying the memory access instruction.

The external server is also a service resource. It can be accessed based on the availability function η. If the external server is available, then the access is defined as a request with input parameters and external server address.

We will continue with specifying a simplified mathematical model of a memory and server.

A. A simplified model of a memory system

A memory is a specific server system. The service resources, in this case, belong to a set of memory locations $\mathcal{M}$. The unique identifier of a memory location is its address α.

The memory can have different resource sets, such as internal registers, and bulk memory. Therefore, the service that a memory is providing needs to make a distinction to which resource a service request is referred to. In this case, the resource identifier τ is used as an input parameter in the service request, besides the address.

In addition, the memory service function should be determined as "store" or "load", by the function id φ.

Definition 4: A *memory system* is a simplified server system with the following constraints to Def. 1:

- α is the address of a memory location, such that $\alpha \in \{0, 1, \ldots, 2^M - 1\}$ in a memory that contains 2^M locations, where M is a positive integer,
- resource id $\tau \in 0, 1, \ldots R - 1$ in a system that uses $R > 0$ resources,
- function id $\varphi \in \{0, 1\}$, where 0 means memory load, and 1 means memory store,
- λ is the input set defined as a set of the address, resource id, and function id, that is $\lambda = \{\alpha, \tau, \varphi\}$,
- ρ is the output value getting a value of performing the output function $\rho = \omega(\alpha, \tau, \varphi)$,
- s_0 is the initial state that represents a *ready* state that waits for an input triple to perform an activity according to the specified function φ. When the service is requested, the state changes to a *busy* state s_1. Once the service response is computed and an output is sent, the state changes to s_0.

If a service is requested while the internal state is $busy$ then the request is held in a queue until the internal state reaches the ready state s_0.

B. A simplified model of a digital service system

The service resources are determined by the state, internal memory and external server resources. Each state $s \in S$ can be determined by the internal registers. The set of all internal registers $\mathcal{R}$ and the set of all memory locations $\mathcal{M}$ determine the internal resources.

Note that in Def. 1 we have defined the values that can be exchanged between the systems. Here we define the locations where these values will be stored and used. Therefore, the set $\mathcal{I} = \mathcal{R} \cup \mathcal{M}$ is a representation of internal resources, as a set of all internal memory locations. Note that these sets should not be mixed with data values μ or the data alphabet Θ.

External resources are determined by a finite number of external servers, with a set of $\mathcal{E}$ memory locations.

Both the Internal and external resources define the service resources.

A service resource may belong to the dew computing concept if it functions both with or without Internet availability. In addition, it needs to be close to the service requestor to belong to a class of dew computing servers.

VI. Discussion

A. Dew Computing Features

Two main features of the dew computing concept will be analyzed in correspondence to our formal specification.

Independence is addressed by the availability function equal to 0 ($\eta = 0$). In our formal specification, the dew server will continue to deliver its services in the case without Internet availability.

Collaboration is addressed by the availability function equal to 1 ($\eta = 1$). In our formal specification, the dew server can exchange information with the cloud servers, which means synchronization of content or control parameters. Also, in this case, the dew server will continue to deliver its services.

Collaboration enables at least the following:

- *upward synchronization* to transfer information to the cloud server,
- *downward synchronization* to transfer information to the dew server, and
- *new service specification* to define new services of the dew server.

Upward synchronization means that the exchange of information with the cloud server is such that data from the dew server is sent to the cloud in order to be available for a wider environment. It will not change the service definition represented by the transformation functions, including transition function δ and output function ω.

Note that the independence feature specifies that the collaboration is not crucial for delivery of services. It means that the collaboration means that the cloud server will receive some information and not have any impact on the performance of the dew server.

The upward synchronization is extensively used in an application of dew computing solutions for IoT, such as wearable eHealth sensors that deliver data to a smartphone, being a dew server, that functions with and without Internet availability. In the case of Internet availability, the smartphone sends all received data to the cloud server.

Downward synchronization means that the exchange of information with cloud servers may change the set of internal memory values μ. Therefore, this will change the internal state of the dew server and initiate results (service output) different from those that will be obtained if the internal state was not changed. For example, this feature is used to define a new content of the dew server that delivers localized web content.

One more function may be used with the collaboration function. It may be used to define a new service with specific transition function δ and output function ω. This makes the dew server a rich environment to deliver services.

B. Comparison with related approaches

Let's compare our server system definition with other definitions and specifications. Zhang et al. [11] specify a service model as a Feedback Control-based Services System. They specify the input of a service consumer and the output as a fulfillment of a service requestor. A specific sensor feeds back the response back in the system for continuous improvement and business transformation. The interior components are service activities/processes, service resources, service information, service people and service partners. In addition, they set the internal goal to increase the profit and decrease the costs, and external goal to reach service level agreement.

In our model, we have identified the service resources and service activities/processes. The service resources are the internal memory and external server resources, and also the service information kept as a state in our definition. The transformation is defined by the transition and output functions that compute the next service information and the output. Our model refers to data computation and does not address people and partners. Also, we do not analyze the business aspects by setting business goals, we describe a digital version of a dew server.

C. Dew computing vs. Edge computing

The essence of edge computing is to push applications, data, and services away from central servers (core) to the edge of a network. It is based on the core-edge topology. While most devices are connected with core-edge topology at the current time, some devices are connected with mesh topologies, such as NYCmesh, Detroit's Equitable Internet Initiative, and eastern Afghanistan's FabFi.

Dew computing is a different approach. It emphasizes its independent operation without the Internet connection and its collaboration with cloud services. Dew computing does not rely on network topology.

Comparing dew computing and edge computing, dew computing is featured by its collaboration with cloud services, and it is not restricted by core-edge topology. Edge computing

also has its advantages. Because currently, core-edge topology is still dominant in the Internet, edge computing encourages researchers and professionals to move applications to the edge of the network, which goes toward the same direction as dew computing.

A cloudlet is a small-scale cloud datacenter located at the edge of the Internet. It is the middle tier of a 3-tier hierarchy: mobile device - cloudlet - cloud. Cloudlet is close to a mobile device but not on the mobile device. On the other hand, dew servers, if introduced, should be on the mobile devices.

Therefore, the formal description of edge computing could be developed in a similar manner as the development of dew computing formal description, but including the network topology.

VII. Conclusion

In this paper, we have introduced a formal specification of the dew computing concept.

Dew computing is based on delivering of (micro) services by a dew server, functioning independent of a wider Internet-based environment, although it can collaborate with other cloud servers in the case of Internet availability.

Our definition of a server that delivers a service is based on a mathematical specification of a Turing machine, adapted to the availability of internal and external resources. In this sense, internal resources are defined as memory that specifies an internal state of the dew server, while the external resources are defined by other services that depend on the availability of the Internet.

Using our introduced formal specification, we have presented examples of simplified systems, including internal memory, and external servers that deliver external services to enable external data to our analyzed service. This completes the formal specification, as it uses a recursive definition of a memory and external servers by its initial definition.

In addition, we have compared edge computing and dew computing with regards to our definition, and indicated the essential difference between dew computing and edge computing.

For future research, the first task would be to fine-tune the definition of a dew server system. We have noticed that one feature of dew server system has not been grasped by Definition 2. This feature is restricted client access to the dew server. Because such feature involves the overall description of the client-server systems, we postpone such discussion to a later time. Future research may also involve the analysis of performance issues that could provide a better insight into our formal specification.

References

[1] Y. Wang, "Cloud-dew architecture," *International Journal of Cloud Computing*, vol. 4, no. 3, pp. 199–210, 2015.

[2] A. Bahtovski and M. Gusev, "Cloudlet challenges," *Proceedia Engineering*, vol. 69, pp. 704–711, 2014.

[3] M. Gusev and S. Dustdar, "Going back to the roots: The evolution of edge computing, an IoT perspective," *IEEE Internet Computing*, vol. 22, no. 2, pp. 5–15, 2018.

[4] Y. Wang, "Definition and categorization of dew computing," *Open Journal of Cloud Computing (OJCC)*, vol. 3, no. 1, pp. 1–7, 2016.

[5] P. P. Ray, "An introduction to dew computing: Definition, concept and implications," *IEEE Access*, vol. 6, pp. 723–737, 2018.

[6] K. Skala, D. Davidovic, E. Afgan, I. Sovic, and Z. Sojat, "Scalable distributed computing hierarchy: Cloud, fog and dew computing," *Open Journal of Cloud Computing (OJCC)*, vol. 2, no. 1, pp. 16–24, 2015.

[7] S. Ristov, K. Cvetkov, and M. Gusev, "Implementation of a horizontal scalable balancer for dew computing services," *Scalable Computing: Practice and Experience*, vol. 17, no. 2, pp. 79–90, 2016.

[8] A. Rindos and Y. Wang, "Dew computing: The complementary piece of cloud computing," in *Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)(BDCloud-SocialCom-SustainCom), 2016 IEEE International Conferences on*. IEEE, 2016, pp. 15–20.

[9] M. Gusev, "A dew computing solution for IoT streaming devices," in *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2017 40th International Convention on*. IEEE, 2017, pp. 387–392.

[10] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Reading Mass, 1979.

[11] L.-J. Zhang, J. Zhang, and H. Cai, *Services computing*. Springer Science & Business Media, 2008.

Enhancing Usability of Cloud Storage Clients with Dew Computing

Tushar S Mane.
TASM2M,
Total Automation Solutions,
Pune, India.
tushar.mane@tasind.com

Himanshu Agrawal.
Department of Computer Science,
Symbiosis Institute of Technology,
Pune, India.
himanshu.agrawal@sitpue.edu.in

Gurmeet Singh Gill.
Brand Manager,
Delicious Tiffin Pvt. Ltd.,
Pune, India.
gurmeet@delicioustiffin.com

***Abstract*— Cloud storage services like Dropbox, Google Drive, Microsoft One Drive have been active and improving unceasingly since 2007. Introduction of desktop/ mobile clients, for example, 'Dropbox Desktop' are the cherry on top, as the users could lever their cloud space from their desktops or mobiles, giving them elegant feel with the file system and storage present on their device. Whatever happens on the device reflects back on cloud space with the corresponding linked account, and vice versa. We can 'partially' say that the computation has been brought down to the ground by such clients. Though this has extremely augmented the user experience, there are still some parts which need to be put in place to complete the picture. This paper highlights some of the missing pieces (offline version management, offline file sharing. and access control list, file URL generation etc.) and their resolutions, which can further improve the usability of the cloud storage clients taking, them towards the completeness. As this is a seamless demonstration of computation happening at device level when there is an intermittent internet connectivity, and then handshaking back with the cloud as it gets connected back to the internet, i.e. Dew Computing, we are referring it as Dewbox. This is one of the many, yet enclosed features of on-device computing, and hence we attempt to encourage researchers to expose possible mechanisms which would utilize the device's potentials to its best. As per the knowledge of authors, this is the first ever attempt till date which traces the missing features of cloud storage clients.**

Keywords—Cloud Computing, Fog Computing, Dew Computing, Cloud Storage, Version Management, Dropbox, Google Drive, One Drive.

I. INTRODUCTION

Storage was the fundamental aspect of Cloud Computing [1] that made it so widespread and beneficial, as it came up with the notions such as Horizontal and Vertical Scaling [2]. Compute services followed the storage services and so do the other services. Latency, security, and privacy are still the foremost concerns of Cloud Computing [3]. To address these issues, Fog Computing [4], which is proximal to devices is presented. Developments in embedded computing [5] have now made devices remarkably powerful. Quantum computing is already knocking the doors [6]. Dew Computing [7] focuses on utilizing device capabilities, especially when those are offline. Dew Computing can be defined as on-device computing, which is not only independent of Cloud Computing but also collaborative with it. Independent means device should work in the absence of internet connectivity, while collaborative meaning whatever happened in absence of internet connectivity is synced appropriately and in order with the cloud as soon as the device gets connected back to the internet. While investigating Storage in Dew (STiD) category [8], we found one of the important missing functionalities, offline version management. Almost 15 GB space is allocated for an unlimited period in free tier. So, students, freelancers, small-scaled or medium scaled organization's developers prefer it as a workspace for their source codes. The purpose is to have cloud as well as a local copy of workspace, so that developers can code online, offline, and at the same time they don't have to worry about version control, which is default feature of cloud storage services. Version management [9] allows one to work freely, without worrying about possible mistakes. One can just switch back to the previous or next version as and when it is necessary. It also offers huge relief on maintenance/ reuse side, one can just restore an applicable version, modify it and get objectives completed. Apart from developers, there are other users (who don't care about versions/ maybe sometimes they do) of storage clients too, who are interested in storing documents, sharing those with any person of choice, on the move. These users can also get benefited by having Dew Computing as an add-on in their storage clients. The sole purpose of the paper is to demonstrate, how missing functionalities of any software client or tool, which are collaborated with the cloud, can be discovered to take them towards completeness, and then unleash the power of devices to achieve the necessary computing i.e. Dew Computing. This is just one of the million possible compute examples to showcase the influence of Dew Computing, and attempt to promote the immense scope in Dew Computing Research.

The paper is structured as follows, second section illuminates Dew Computing to its state of art in a comprehensive manner. The third section enlightens the issues in the current cloud storage clients with the example of Dropbox, while the fourth section proposes the extension to 'Storage in Dew (STiD)' category of Dew Computing to resolve the issues in the current model of cloud storage clients. In the same section, we essentially propose the architecture of Dewbox. Please note implementation guidelines of only the missing features are given. We conclude the paper in the fifth section along with future directions.

II. DEW COMPUTING: STATE OF ART

Current Dew Computing Research revolves around four principal visions by various researchers involved in Dew Computing. However, all forks joining at one common feature, on-device computing which is collaborative with upper layers of computing. Associate Professor, Dr. Yingwei Wang, School of Mathematical and Computational Sciences, University of Prince Edward Island, Canada, states it as a computing residing on the 'on premise' computer, which is independent of cloud in offline mode, while collaborative with the cloud in case of online mode [8]. Whatever happened during offline mode would be synchronized and correlated back with the cloud in the subsequent online

mode. The following diagram (fig. 1) depicts the proposed Cloud-Dew Architecture, wherein any device in the local network will be served by corresponding Dew Server. Devices can still avail minimal set of services or frequently used functions from the Dew Server for unbroken computing.

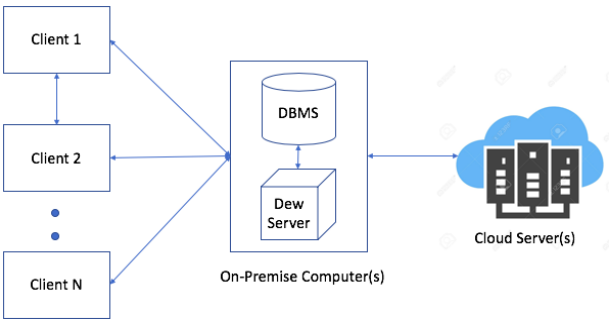


Fig. 1. Cloud- Dew Architecture.

The key objective is to facilitate with the services to users even if there is no internet connection. The use case has been classily demonstrated with the category Web in Dew (WiD) [10], wherein user can still access the website in absence of network connectivity. Key set of minimal functions are still served by 'on premise' server with the help of application, web, and database server running on it. Example, you can browse your Facebook posts and pictures in your spare time even if you don't have an active internet connection. Further, Dr. Yingwei Wang has categorized this generic architecture into several type of services, so that parallel research work can be started for rapid growth of Dew Computing. In this paper we explore Storage in Dew (STiD).

Second research involvement on Dew Computing is directed by Dr. Karolj Skala, Professor at Rudjer Boskovic Institute, Zagreb, Croatia. He proposes Dew Computing to be Context as a Service (CaaS) to offload Cloud Computing [11]. Context as a Service (CaaS) involves processing data at ground, and provide a meaningful context to the cloud. This will surely be a helping hand to the cloud servers. This would scale computing power drastically (fig. 2) and will open doors to solutions to the various computing problems which were considered to be hard to solve till now.

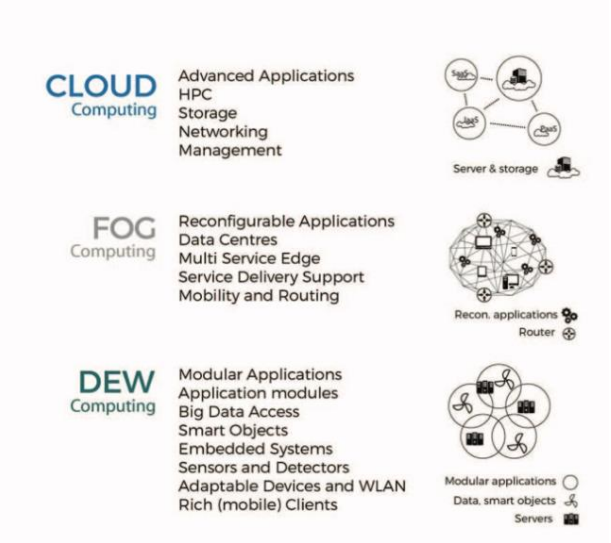


Fig. 2. Scalable Computing Hierarchy [11].

Going forward, the professor has coined the terms Distributed Information Service Environment (DISE), Global Information Processing Environment (GIPE) and Low Power Low Information Processing (LPLIP) as agents of massively distributed and connected physical things.

Dr. Sasko Ristov, Institute of Computer Science, University of Innsbruckalso, has similar vision of utilizing maximum resources at the roots, for information processing, before computation is handed over to the cloud. Researcher proposed 'computation scalability mechanism and its load balancing' in his research work [12].

Recently, Mr. Partha Ray extended Dr. Yingwei Wang's Cloud-Dew Architecture and introduced some terms to support the extended model [13]. He aims to have lightweight 'Dew Server' on client itself, which should serve one client at a time, and store most frequently used functions. In case of data loss, the 'Dew Server' should be able to recover from cloud server from the last checkpoint. Local copy of data should be as small as possible which is referred as 'Dew Site'. The refinement consists of how 'Dew Site' can be modified by 'Dew Client'. 'Dew Script' is a web script file, which will be used for modification of 'Dew Site'. These modifications will be supervised by a 'light weight web come application controller' called 'Dew Analyzer', which will be responsible for maintaining the 'Dew Site' state in respective database(s) present in Database Management System on the 'Dew Server'. The operational details are made clear by 1:1 and 1: N mappings between 'Dew Server' and 'Dew Site' as shown in fig. 3.

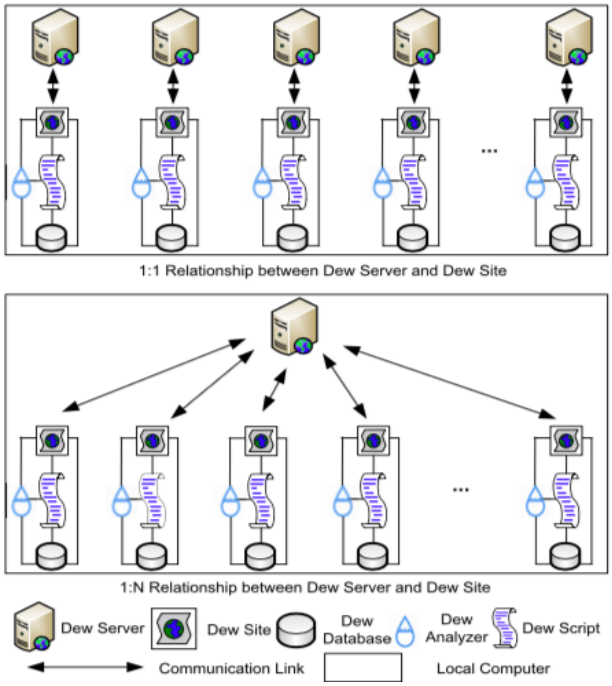


Fig. 3. Mappings Between Dew Server(s) and Dew Site(s) [13].

Last but not the least, in our (me and Dr. Himanshu Agrawal) recent paper [14], which clubs Cloud-Fog-Dew Computing paradigms in to one Service Computing Ecosystem, we shown how Dew Computing can considerably reduce the computational latency. One of the key properties of Fog Computing is its heavy geographical distribution to support the scalability [15]. This geo-graphical distribution comes with the maintenance overload

and hence the possible computing outage. So we put forward a 'Dew Node Architecture', which enables 'Dew Node' to be a Service Provider or Consumer. Fig. 4 shows architecture of 'Dew Node'.

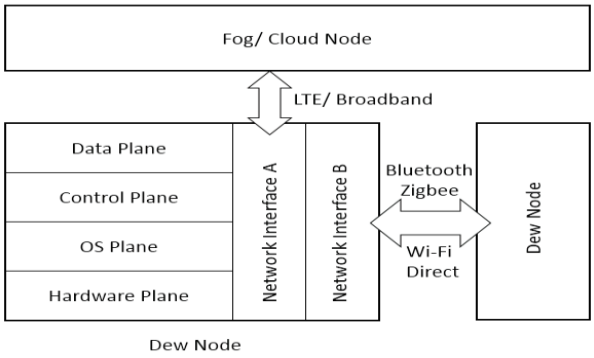


Fig. 4. Dew Node Architecture [14].

So, in case Fog Computing layer fails to provide service to the end devices, one end node, which now becomes 'Dew Node' will provides service to other end node i.e. 'Dew Node' with contextual intelligence embedded in them.

III. Cloud Storage Clinets: The Missing Elements

As stated before, this paper emphasis on Storage in Dew (STiD) class of Dew Computing. Existing functions of cloud storage clients are well versed, well established, and stable. But, there are some major missing features, which we trace in this section. Please note very carefully, that we have chosen Dropbox for demonstration purpose, as Dropbox is implemented entirely using open source technologies. However, same issue can be reproduced on Google's backup and Sync (Previously Google Drive Desktop) and Microsoft's One Drive and any other cloud storage client.

A. Version Management

We illustrate the conflict between local and cloud versions lucidly with the help of small experiment done on Dropbox. Please note the folder structure, text editor and symbols. File is stored in Dropbox folder, the special space created on user's device after installing Dropbox client. We have used 'nano' editor, but any text editor would do. Green tick indicates local copy is in sync with cloud, while cycle symbol indicates, it not synced yet (or syncing under progress).

User creates file 'DewBox.c' on his device (in Dropbox folder, as shown in fig.5) and it is synced with its cloud space as there is active internet connection. (Version 1, On-device)

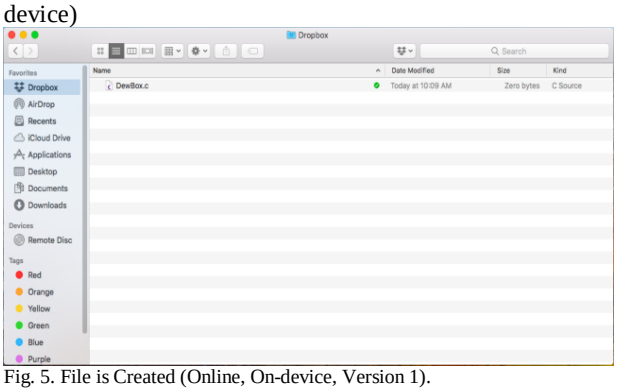


Fig. 5. File is Created (Online, On-device, Version 1).

Instantly, cloning happens on Dropbox cloud with your linked account (Version 1- Cloud), fig. 6.

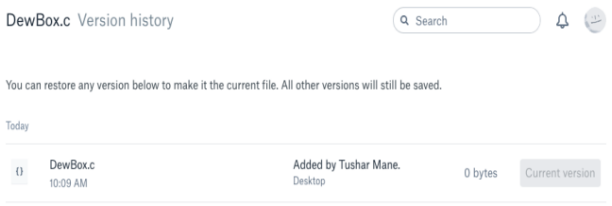


Fig. 6. File Create Operation Clonned (Cloud, Version 1).

Program description, Author info, and Date is added in program while online, file is saved and closed (Meta Data Added- Version 2, On-device), fig. 7.

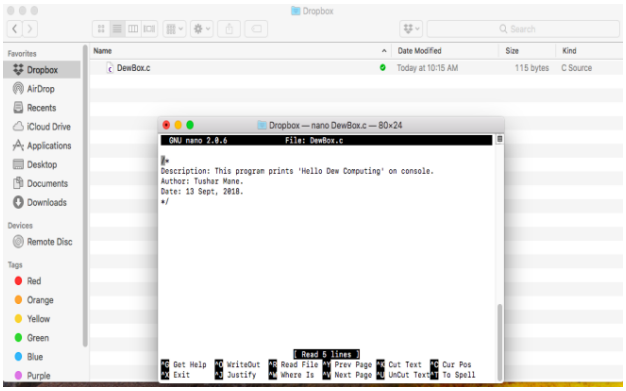


Fig. 7. Meta Data is Added in File (Online, On-device ,Vesrion 2).

With internet connection available, changes are reflected on respective cloud space (Version 2- Cloud)- Fig. 8

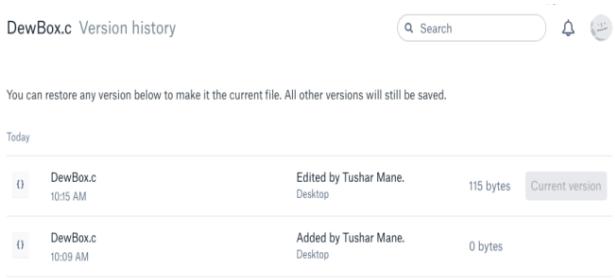


Fig. 8. Meta Data Reflected (Cloud, Version 2).

Now, internet connection is disconnected deliberately and local file is added with some statements and saved, fig. 9. (Version 3, On-device)

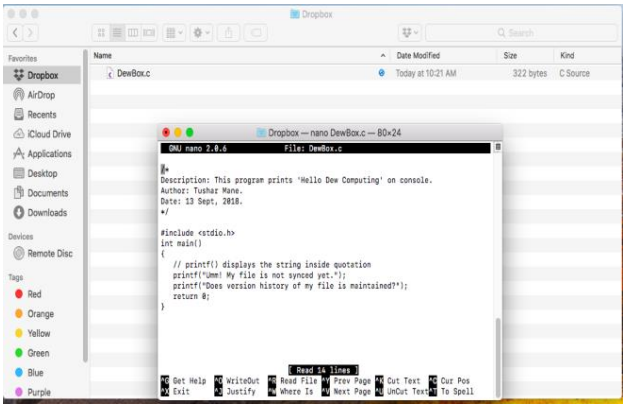


Fig. 9. Statements are Added in File (Offline, On Device, Version 3).

Again, without internet connection, statements in the file are modified and changes are saved, fig. 10. (Version 4, On-device)

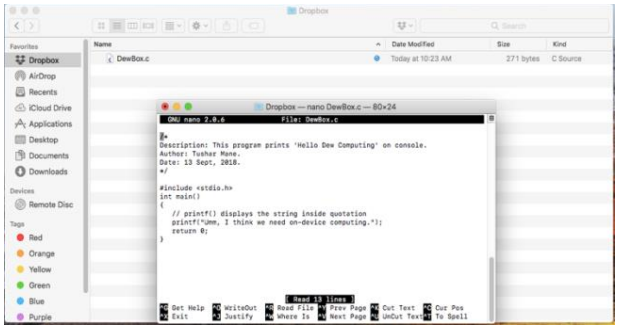


Fig. 10. Some more Modifications in file (Offline, On-device, Version 4).

Now, file is modified again and internet is enabled. (Version 5, On-device)- fig. 11.

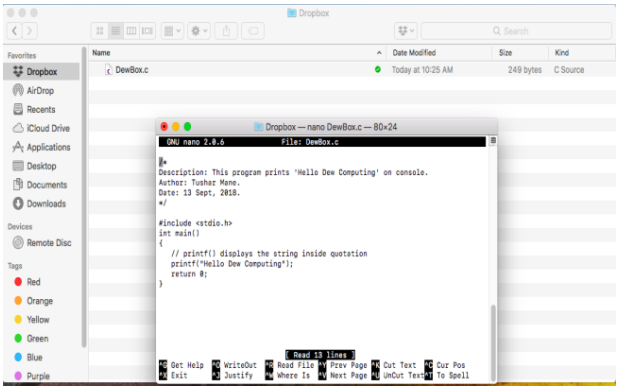


Fig. 11. Some Chnages are made and then Connected to Internet (Vesrion 5).

Immediately, file is synced with cloud and digital twin is created as shown in fig. 12 (Version 5).

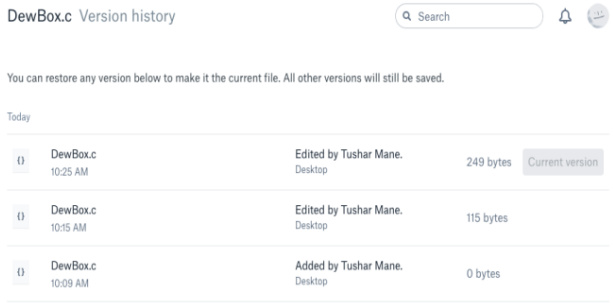


Fig. 12. File gets Synced (Cloud, Version 3).

Here's the serious problem. What about Version-3 and Version-4? If student/ developer/ freelancer relies on 'anytime & anywhere' feature of cloud, and does some work at home (or on another device). Next day he goes to office (or changes the device), and now wants to restore the program back to Version-3 or Version-4, how can it be done? Versions which were created, when he was unknowingly not connected to the internet are lost, right? Now, this is a very small and simple example (just for demonstration). In real world there are dependencies among files/ modules, critical functionalities, a lot of automatic documentation, and many more factors are involved. This is severe missing part of a system. Now let's have a look at possible add-ons.

B. Enhanced File Operations (File sharing, File Link Generation, Access Control List etc.)

Roaming people (sales and marketing, business development, site support, to name a few) or even other people/ students, who frequently need to share the documents, keep adding (or removing) people to (or from) shared list (Access Control List) or share a public file URL, might not always have an active internet connectivity, may be due to absence of network, low bandwidth or end of mobile data quota. In such situations most of the people generally tend to forget completing this to-do list. Fig. 13. shows current online operations users can do via client.

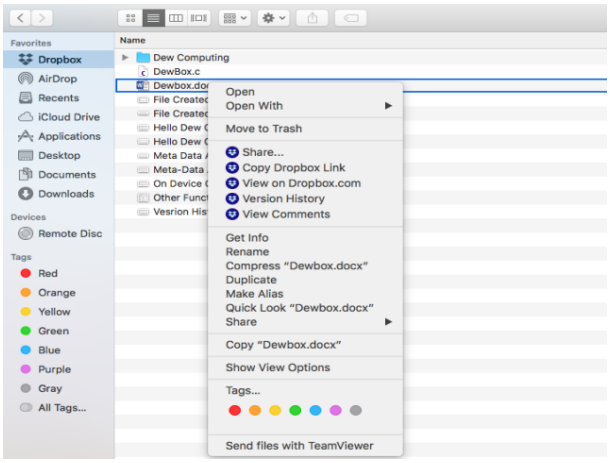


Fig. 13.Current Options in Cloud Storage Clients (e.g. Dropbox)

So, an add-on which enables these pending tasks to be queued in offline mode, so that they can just do these kinds of activities on the move, without having to maintain this to-do list in their mind or a diary, would remarkably increase the usability of cloud storage clients.
Next section explains, Dewbox architecture/ implementation guidelines and how it can overcome above issues.

IV. DEWBOX: TOWARDS COMPLETENESS OF CLOUD STORAGE CLIENTS

As mentioned before, all current features of cloud storage clients are well established, well versed, and stable. So architecture and implementation guidelines of Dewbox only focuses on offline version management, file sharing, URL generation and access control list.

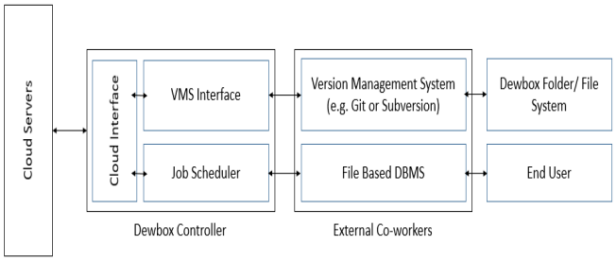


Fig. 14. Proposed Architecture for traced Missing Functionalities/ Add-ons

Architecture, fig. 14, mainly consists of two components, 'Dewbox Controller' and 'External Co-workers'. 'Dewbox controller' contains a component called 'VMS (Version Management System) Interface', which pulls out all the version history of any file from the external version

management system (e.g. Git or Subversion) [16] [17] and shares it with cloud storage server, with the help of 'Cloud Interface', fig. 15, whenever you get connected to the internet. So a fine grained version history of any file or set of files can be maintained. Open source version management systems like Git or Subversion would be installed along with Dewbox, and VMS interface will trigger repository initialization, querying the repository and deletion of the repository automatically in the background, based upon file operations done by user.

Fig. 15. Handshaking between Cloud Interface Module and Cloud Server.

Add-ons like 'Offline File Sharing, Link Generation and Access Control List' are taken care by the second component of 'Dew Controller', the 'Job Scheduler'. 'Job Scheduler' is Simple Queue Data Structure Implementation for which data elements are filled from the file-based Database Management System like SQLite [18]. Whatever operations are scheduled in offline mode by user, are put in a table called 'job_queue'. This table would be queried by 'Job Scheduler' to populate the job queue.

There would be three other tables in database, one for storing contact list of the linked account, other for list of files and their access control, and lastly for storing file URLs and access control. Whenever user says, I want file 'A' to be shared with person 'XYZ' with 'Read-Only' permission, a job will be created by selecting contact 'XYZ' from 'contact_list' table, file 'A' from 'files' table, inserting them in a 'job_queue' table with operation as 'share' and permission as 'Read Only'. 'Job scheduler' reads this table, put the jobs in queue and whenever device gets connected to the internet, it hands over this queue to the cloud server via 'Cloud Interface'. 'Cloud interface' packs these jobs in such a manner that those would be understandable by 'Cloud Server(s)'.

V. Conclusion and Future Directions

Cloud Storage has become prevalent with Cloud Storage Clients, which allows users to do file operations on their devices, irrespective of file system, even in absence of network connectivity. Operations done in offline mode are later get synced with the linked cloud storage. This phenomenon perfectly fit in with definition of Dew Computing, which states, computing which is independent and collaborative with cloud. Hence, Cloud Storage Clients such as Dropbox Desktop, Google Drive Desktop (Now Back up and Sync), Microsoft's One Drive, are categorized under Storage in Dew (STiD) class of Dew Computing. But the critical function of version management, which could lead to some serious issues, was still absent. Also, usability of such clients can also be increased by the offline extensions like File Sharing, URL Generation and Access Control List. In this paper, we attempt to trace these missing features and add-ons. We also suggest an architecture and implementation guidelines for the same. We name it as Dewbox, which would take Cloud Storage Clients towards completeness.

The only purpose behind the paper is to demonstrate how various software or hardware clients or tools, which are connected to the cloud, can be made more independent and collaborative by utilizing resources present on them for increasing their usability. Dew Computing is surveyed in a broad manner to appeal researchers to come forward and contribute in this growing area, which will drastically offload cloud servers and provide seamless computing even in unexpected interruptions in Fog Computing layer.

We highly anticipate implementation of this paper as a foremost future work. We have not focused on security portion of Dewbox, which could mean, opening the Dewbox folder with credentials or securing file-based database system which we have recommended or more. We highly encourage to explore hardware clients like Arduino and other embedded boards for Dew Computing Research as Internet of Things, Quantum Computing are already here and we want Dew Computing to be essential bit of those.

References

[1] Yuhang Yang and Maode Ma, "A Survey of Cloud Computing," Proceedings of the 2nd International Conference on Green Communications and Networks 2012 (GCN 2012), Volume 3. pp 79-84. 2013.

[2] Chien-Yu Liu, Meng-Ru Shie , Yi-Fang Lee, Yu-Chun Lin and Kuan-Chou Lai, "Vertical/Horizontal Resource Scaling Mechanism for Federated Clouds," International Conference on Information Science & Applications (ICISA), 2014.

[3] Danilo Ardagna, "Cloud and Multi-cloud Computing: Current Challenges and Future Applications," IEEE/ACM 7th International Workshop on Principles of Engineering Service-Oriented and Cloud Systems, 2015.

[4] Mohammad Aazam and Eui-Nam Huh, "Fog computing: The Cloud-IoT/IoE middleware paradigm," IEEE Potentials. May/June, 2016.

[5] Jing Huang, Renfa Li, Jiyao An, Derrick Ntalasha, Fan Yang and Keqin Li, "Energy-Efficient Resource Utilization for Heterogeneous Embedded Computing Systems," IEEE Transactions on Computers, 2017, Volume: 66, Issue: 9.

[6] Charles Day, "Quantum Computing Is Exciting and Important--Really!," Computing in Science & Engineering, 2007, Volume: 9, Issue: 2.

[7] Andy Rindos, Yingwei Wang, "Dew Computing: The Complementary Piece of Cloud Computing," IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom), 2016.

[8] Yingwei Wang, "Definition and Categorization of Dew Computing," Open Journal of Cloud Computing (OJCC), Volume 3. Issue 1, 2016.

[9] B. Westfechtel, B.P. Munch and R. Conradi, "A layered architecture for uniform version management," IEEE Transactions on Software Engineering, 2001, Volume: 27, Issue: 12.

[10] Yingwei Wang and David Leblanc, "Integrating SaaS and SaaP with Dew Computing," 2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom), 2016.

[11] Karolj Skala and Davor Davidovic, "Scalable Distributed Computing Hierarchy: Cloud. Fog and Dew Computing," Open Journal of Cloud Computing (OJCC) Volume 2, Issue 1, 2015.

[12] Sasko Ristov, Kiril Cvetkov, and Marjan Gusev, "Implementation of a Horizontal Scalable Balancer for Dew Computing Services," Scalable Computing: Practice and Experience, 2016, Volume 17, Number 2, pp. 79-90.

[13] Partha Pratim Ray , “An Introduction to Dew Computing: Definition, Concept and Implications," IEEE Access, 2018, Volume: 6.

[14] Tushar S Mane and Himanshu Agrawal, “Cloud-fog-dew architecture for refined driving assistance: The complete service computing ecosystem”, IEEE 17th International Conference on Ubiquitous Wireless Broadband (ICUWB), 2017.

[15] Amir Vahid Dastjerdi and Rajkumar Buyya, “Fog Computing: Helping the Internet of Things Realize Its Potential,” IEEE Computer, 2016, Volume: 49, Issue: 8.

[16] https://git-scm.com/

[17] https://subversion.apache.org/

[18] https://www.sqlite.org/index.html

Overview of Cloudlet, Fog Computing, Edge Computing, and Dew Computing

Yi Pan*, Parimala Thulasiraman†, Yingwei Wang‡
*Geogia State University, USA
yipan@gsu.edu
†University of Manitoba, Canada
thulasir@cs.umanitoba.ca
‡University of Prince Edward Island, Canada
ywang@upei.ca

***Abstract*—Cloudlet, Fog Computing, Edge Computing, and Dew Computing are post-cloud computing models. Researchers and public need to grasp the essential meaning of each one and their differences. In this talk summary, we will describe the origins, definitions, basic principles, and applications of these computing models.**

***Index Terms*—Dew computing; Fog computing; Edge computing; Cloudlet; Cloud computing; Network topology; Internet of Things; Mobile applications; Blockchain.**

I. Introduction

Cloudlet, Fog Computing, Edge Computing, and Dew Computing are computing models proposed to provide some features that Cloud Computing cannot provide. They share one common feature: they all perform computing tasks at devices that are closer to users. We may call them post-cloud computing models. Researchers and public need to know their characteristics, to know their similarities and differences. We would like to provide an overview to these computing models.

These post-cloud computing models cover huge amount of research work. We do not intent to provide a full survey to the whole landscape of these computing models in this talk. We only concentrate on the following aspects of each model:

1) the origin: when did it start and how it was started;
2) the definition: what does it mean;
3) the principles and applications: how does it work and how was it used.

We would like to explain our positions regarding to the origins and definitions of these models.

For origins, in our understanding, every computing model goes through the following steps for its origination:

1) Before the concept was proposed, some concrete technical approaches that are very similar to the new concept or exactly the same with the new concept were proposed as research ideas and/or applied in products or services;
2) The new concept was proposed after technical accumulation;
3) After the concept was proposed, technical approaches based on the new concept were widely and quickly spread; existing approaches were interpreted with the new concept; new approaches were proposed according to the new concept.

No computing model can be proposed without technical accumulation described in Step 1. A long-term accumulation process is necessary for the establishment of a computing model.

The origination of a new concept is a significant event because the new concept leads researchers to explore solutions to wide range of problems using a paradigm or a framework that comes with the new concept. Thus, we would like to introduce the origin of each computing model.

For definitions, each computing model may have more than one definition. Different researchers may have different opinions toward these definitions. For each computing model, we try to find a definition that, we believe, accurately describe this model.

As a general statement, this talk summary was prepared for tutorial and discussion purposes. It reflects our limited knowledge and subjective opinions; we do not guarantee its accuracy and completeness, although most of our descriptions have supporting references.

II. Cloudlet

A. Origin

Although the word *cloudlet* existed long time ago with different meanings, it was started being used in the meaning of a computing arrangement in 2009 [1][2].

B. Definition

The following is a definition of a cloudlet [2]:

A cloudlet is a trusted, resource-rich computer or cluster of computers that is well-connected to the Internet and is available for use by nearby mobile devices.

C. Principles and Applications

The Cloudlet model promotes to put small-scale cloud data centers at the edge of the Internet. A cloudlet is the middle tier of a 3-tier hierarchy: mobile device - cloudlet - cloud. A cloudlet is close to a mobile device but not on the mobile device.

III. Fog Computing

A. Origin

Fog Computing was proposed by Cisco. It was first proposed by Flavio Bonomi, Vice President of Cisco Systems, in a keynote presentation at a conference in Sept. 2011 [3][4].

B. Defintion

The following is a definition of Fog Computing [5]:

Fog Computing is a scenario where a huge number of heterogeneous (wireless and sometimes autonomous) ubiquitous and decentralised devices communicate and potentially cooperate among them and with the network to perform storage and processing tasks without the intervention of third-parties. These tasks can be for supporting basic network functions or new services and applications that run in a sandboxed environment. Users leasing part of their devices to host these services get incentives for doing so.

C. Principles and Applications

Fog Computing extends Cloud Computing and services to devices such as routers, routing switches, multiplexers, and so on. It mainly involves automation devices because Fog Computing was proposed with Internet of Things (IoT) as its background.

IV. Edge Computing

A. Origin

The term *edge cluster* was used in a paper in August 2015 [6]. *Edge Computing* was proposed for the first time in October 2015 [7]. Some work has been done before this time. As discussed in Section I, we consider those work as the accumulation work before its birth.

A paper used the term "computing on the edge" in 2004 [8], but it is an "early flavor of edge computing" and the new vision of Edge Computing was "far beyond this initial approach" [7]. The fact that the accumulation work did not use this term also indicates that this paper was not the origin of Edge Computing.

Many research papers about Edge Computing appeared after 2015. It is reasonable to say that Edge Computing was originated in 2015.

B. Definition

The following is a definition of Edge Computing [9]:

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. Here we define edge as any computing and network resources along the path between data sources and cloud data centers.

C. Principles and Applications

Edge Computing pushes applications, data, and services away from central servers (core) to the edge of a network; it is based on the core-edge topology [9][10].

Cloud offloading, video analytics, smart home / smart city are some examples where Edge Computing can be actively applied to [11][12].

V. Dew Computing

A. Origin

Dew Computing was proposed in 2015 [13][14][15]. The first paper became online in January 2015.

B. Definition

The definition of Dew Computing can be found in [16]:

Dew Computing is an on-premises computer software-hardware organization paradigm in the Cloud Computing environment where the on-premises computer provides functionality that is independent of cloud services and is also collaborative with cloud services. The goal of Dew Computing is to fully realize the potentials of on-premises computers and cloud services.

C. Principles and Applications

Dew Computing is a new computing model appeared after the wide acceptance of Cloud Computing. While Cloud Computing uses centralized servers to provide various services, Dew Computing uses on-premises computers to provide decentralized, cloud-friendly, and collaborative micro services to end-users.

Dew Computing is complementary to Cloud Computing. The key features of Dew Computing are that on-premises computers provide functionality independent of cloud services and they also collaborate with cloud services.

VI. Comparison and Discussion

A. Similarities

All these computing models share a common feature: they all perform computing tasks at devices that are closer to users. It is hard to determine the exact differences among these models by checking their definitions. The reasons are:

1) Normally a computing model was proposed to solve a specific problem with a narrow definition. With the progress of research, researchers tend to expand the definition to cover a wider range of area. Thus the definitions of these computing models become quite similar. Such definition expansion reflects researchers' eagerness and excitement in exploring new technologies.
2) Even if differences among these models are found in definitions, some researchers may have different opinions to these definitions.

To understand the underlying reasons of these similar computing model definitions, we had better take a bird's view position to observe the general trend in the history of computer science. Dr. Mahadev Satyanarayanan [10] summarized the past history in the following quote:

"Since the 1960s, computing has alternated between centralization and decentralization. The centralized approaches of batch processing and timesharing prevailed in the 1960s and 1970s. The 1980s and 1990s saw decentralization through the rise of personal computing. By the mid-2000s, the centralized approach of cloud computing began its ascent to the preeminent position that it holds today. Edge Computing represents the latest phase of this ongoing dialectic."

After the widely acceptance and huge success of Cloud Computing, some researchers discovered the limitations of Cloud Computing and proposed remedial solutions from different perspectives. Not only Edge Computing, other models such as Cloudlet, Fog Computing, and Dew Computing were also proposed as the result of this trend.

B. Differences

Although these computing models reflect the same trend in response to Cloud Computing's limitations, these models were quite different because:

1) they originated from different background;
2) they were proposed to solve different problems;
3) they are related to different disciplines or industries;
4) they deal with different types of devices and environment;
5) they have different methodologies. Here we would like to point out some differences among these models.

Cloudlet features micro data centers; it is related to mobile services. Micro data centers could be set up by mobile service providers, application providers, or even users.

Fog Computing is tightly related to Internet of Things. Fog Computing emphasizes proximity to end-users and client objectives, dense geographical distribution and local resource pooling, latency reduction and backbone bandwidth savings.

Edge Computing's rational is that computing should happen at the proximity of data sources [9]. Edge Computing is also tightly related to IoT.

Dew Computing is more closely related to software design; its strong point is to inspire novice applications. Dew Computing was proposed to solve the data availability problem when an Internet connection is not available. Dew Computing's features, categories, and architecture are helpful for new applications be developed. Dew Computing normally does not involve edge devices such as routers and switches.

Sometimes, the difference is quite clear. For example, if Cloudlet model is introduced in mobile applications, a 3-tier hierarchy: mobile device - cloudlet - cloud would be established. A cloudlet is close to a mobile device but not on the mobile device. If Dew Computing is introduced, the dew component would be on the mobile devices.

Different models may work together. For example: A hierarchy was proposed [15] for Cloud Computing, Fog Computing, and Dew Computing to work together.

Different models may obtain similar results. For example, an Edge Computing idea about cloud/edge applications [17] has similar ideas with the cloud-dew architecture proposed in Dew Computing [13].

Each model may have its special strength. For example, the Dewblock system [18], that small-data-size blockchain clients with full node features, can hardly be classified into Cloudlet, Fog Computing, or Edge Comptuing applications; it is only possible under the computing model of Dew Computing.

C. Choice Suggestions

If someone is interested in these post-cloud computing models, which one should he/she choose? What should be considered in making a choice? Here we give some suggestions.

If you are interested in improving mobile services, from services providers' viewpoint or from application developer's viewpoint, Cloudlet model is the suitable model for you to work on.

If you are related to IoT research or IoT industry, Fog Computing is the area you should pay attention to. With the development of IoT, huge amount of sensors will be deployed everywhere. The best place for computing powers to process data from these sensors should not be far away cloud servers or low-capacity sensors. Devices such as routers and switches are a better choice.

If you are interested in infrastructure design, such as smart home / smart city, or are interested in cloud offloading for improved efficiency, Edge Computing could be a suitable choice.

If you are interested in the design of novice distributed applications, Dew Computing could bring you with inspirations and architectural assistance. Dew Computing normally does not involve edge devices, such as routers and switches; Dew Computing is not restricted by network topology.

VII. Conclusion

Cloudlet, Fog Computing, Edge Computing, and Dew Computing spire in the post-cloud world. They were proposed to solve different problems. They involve different devices. They have different methodologies. They have only one belief in common: Cloud Computing should not be the only form of computing. The essential differences among them are not in their definitions that claim their coverages because definitions can be easily updated, expanded, and interpreted in different ways. The essential values of these computing models exist in their built-in principles, architectures, styles, and philosophy. Similar to programming languages, although each programming language has full computing power of a Turing Machine, each language has its own style, strength, and characteristics. These computing models will provide different frameworks, paradigms, guidelines, and architectures to researchers and developers in the post-cloud era.

References

[1] S. Ibrahim, H. Jin, B. Cheng, H. Cao, S. Wu, and L. Qi, "CLOUDLET: towards mapreduce implementation on virtual machines," in *Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing, HPDC 2009, Garching, Germany, June 11-13, 2009*, 2009, pp. 65–66. [Online]. Available: http://doi.acm.org/10.1145/1551609.1551624

[2] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct 2009.

[3] Flavio Bonomi. (2011, Sept.) Connected vehicles, the internet of things, and fog computing. [Online]. Available: https://www.sigmobile.org/mobicom/2011/vanet2011/program.html

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12. New York, NY, USA: ACM, 2012, pp. 13–16. [Online]. Available: http://doi.acm.org/10.1145/2342509.2342513

[5] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014. [Online]. Available: http://doi.acm.org/10.1145/2677046.2677052

[6] Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, P. Bahl, and I. Stoica, "Low latency geo-distributed data analytics," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 421–434, Aug. 2015. [Online]. Available: http://doi.acm.org/10.1145/2829988.2787505

[7] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 37–42, Sep. 2015. [Online]. Available: http://doi.acm.org/10.1145/2831347.2831354

[8] M. Rabinovich, Z. Xiao, and A. Aggarwal, "Computing on the edge: A platform for replicating internet applications," in *Web Content Caching and Distribution*, F. Douglis and B. D. Davison, Eds. Dordrecht: Springer Netherlands, 2004, pp. 57–77.

[9] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct 2016.

[10] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan 2017.

[11] Ganesh Ananthanarayanan and Victor Bahl and Alec Wolman. (2008, Oct.) Edge computing. [Online]. Available: https://www.microsoft.com/en-us/research/project/edge-computing/

[12] G. Ananthanarayanan, P. Bahl, P. Bodk, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha, "Real-time video analytics: The killer app for edge computing," *Computer*, vol. 50, no. 10, pp. 58–67, 2017.

[13] Y. Wang, "Cloud-dew architecture," *International Journal of Cloud Computing*, vol. 4, no. 3, pp. 199–210, 2015.

[14] Y. Wang and Y. Pan, "Cloud-dew architecture : realizing the potential of distributed database systems in unreliable networks," in *Proceedings of the 21st International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA15)*, Jul. 2015, pp. 85–89.

[15] K. Skala, D. Davidovic, E. Afgan, I. Sovic, and Z. Sojat, "Scalable distributed computing hierarchy: Cloud, fog and dew computing," *Open Journal of Cloud Computing (OJCC)*, vol. 2, no. 1, pp. 16–24, 2015.

[16] Yingwei Wang, "Definition and categorization of dew computing," *Open Journal of Cloud Computing (OJCC)*, vol. 3, no. 1, pp. 1–7, 2016.

[17] Julia White. (2018, Sept.) Microsoft azure enables a new wave of edge computing. heres how. [Online]. Available: https://azure.microsoft.com/en-us/blog/microsoft-azure-enables-a-new-wave-of-edge-computing-here-s-how/

[18] Y. Wang, "Dewblock: A blockchain system based on dew computing," in *Proceedings of The 3rd International Workshop on Dew Computing*, Oct. 2018, pp. 0–0.

IEEE DewCom STC

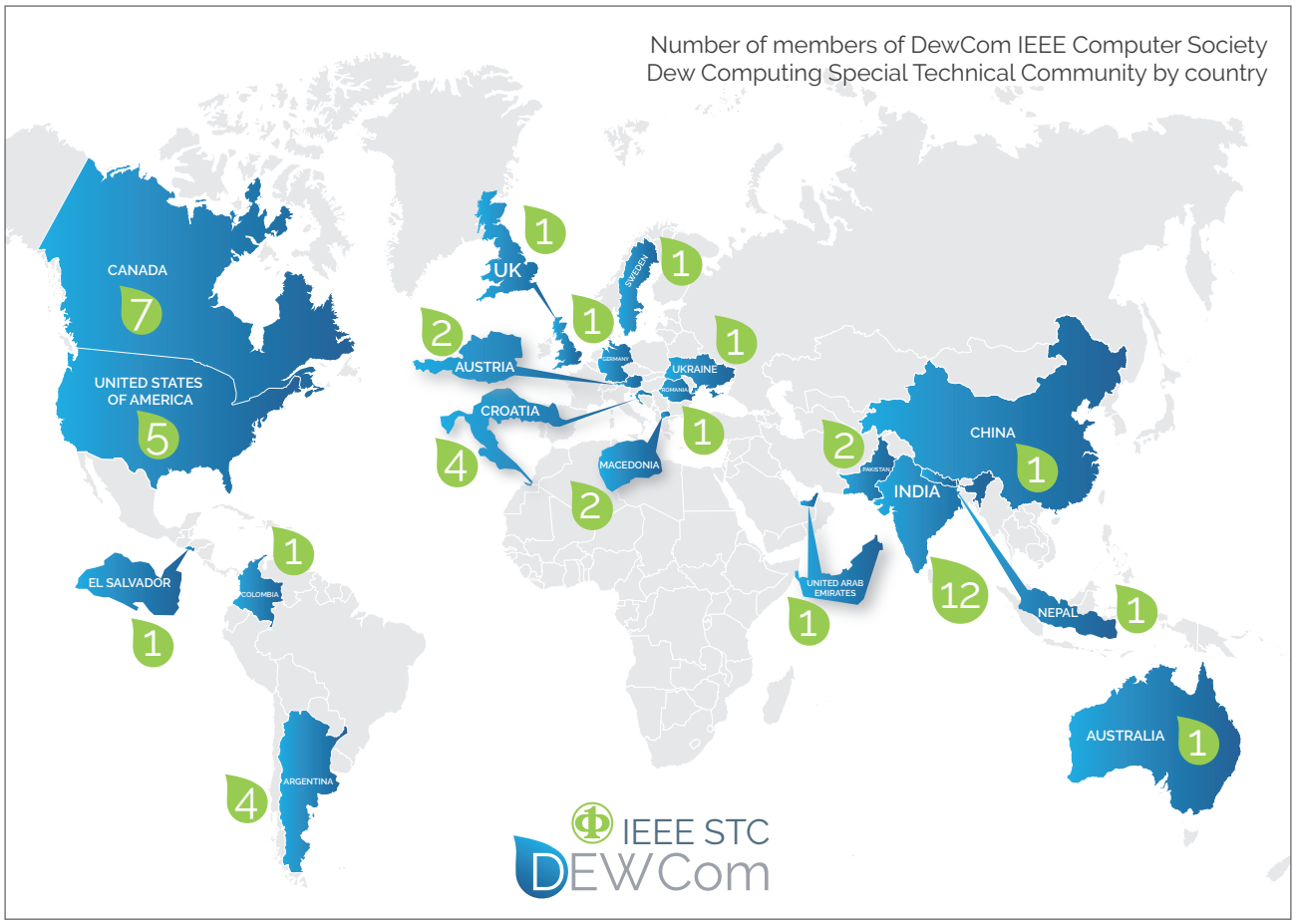
IEEE Computer Society Dew Computing Special Technical Community (IEEE DewCom STC) is a new worldwide Open Community and forum for researchers, professionals, and students in the area of Dew Computing.

Dew Computing is an emerging research/application area that is the complementary piece of cloud computing. The goal of Dew Computing is to fully realize the potentials of on-premises computers and cloud services. The vision of the Dew Computing Special Technical Community is that its efforts shall facilitate dew-computing research and dew computing application, for the benefit of all users and providers of the future global cloud-dew computing environment.

IEEE STC DEWCom

So far, 49 members from 19 countries have joined together.
Join yourself if you want to contribute to the development of this new paradigm

https://stc.computer.org/dewcomputing/



The Rainbow Global Service Ecosystem

Karolj Skala
Ruđer Bošković Institute
Zagreb, Croatia
skala@irb.hr

Zorislav Šojat
Ruđer Bošković Institute
Zagreb, Croatia

Abstract—**A Rainbow is a complex service ecosystem of interdependent components of Cloud–Fog–Dew Computing paradigm layers that all work together to enable a seamless system of global services.**
This paper widely and freely (technological and the Philosophical) considers visions and perceptions in order to liberate conceptual scintillation or imagination. The main aim is not to give strict solutions, but to point towards the extreme broadness of present day Computer Science and computer/digital electronics usage, to point towards some possible future development avenues, and to give a simple analogy as a broad conceptual systematical overview, but including some concrete architectural guidelines. The paper presents the IEEE Dew Computing Special Technical Community (IEEE DewCom STC) as a virtual scientific-research and development environment for Dew Computing platform and application development as well as a collaboration model consideration.

Keywords—Cloud computing, Fog Computing, Dew Computing, Rainbow Service, serveware

I. INTRODUCTION

There are generally two main questions, or better to say problems, to be answered and solved in Computer Science. The one often tackled is "How do we do it?". In this area our civilization is in early stage, as much effort is put into imagining gadgets of all sorts and then solving the "Hows".
However, a much more rarely asked question indicates a big generic problem, spreading from philosophy to psychology and from societal behavior to the very physical – the soil we live on and the air we breathe. This over important and often under regarded question is actually "What do we want to achieve as a civilization?" "What is our final goal?" "What kind of human living environment do we wish?" "What are our civilization aims?"
It is the direct responsibility of us Computer Scientists and of the Computer Science itself to involve itself in the philosophy and ethics of what we do. It is also our direct responsibility to find proper means of education of future generations towards the vast possibilities of proper, ethically and philosophically correct computer usage for the benefit of not only humankind, but also of our own planet and its environment. We may not forget that "empathy", "ethics", "love", "heart", "soul" are notions without which we Humans would not be able to live properly in a civilized society, together with "imagination" and "intuition", and that all those properties are not existing in the computer hardware/software we develop. Will we ever be able to properly "programme" love or empathy or ethics or intuition into future computers? But without those prerequisites we have to be extremly careful of how much power over our own lives and our living environment we give to pure "technical" solutions.
Consequently in addition to a lot of "hows", we also have to solve a lot of "whys"! Or, actually, presently it is the moment to put much more intellectual effort in the "whys".
So, actually there are two major aspects of future development of Computer Science: The Technological and the Philosophical. In this article we will take a slightly unconventional approach, using Cybernetic principles, to shed some light on the overall area which in present day must be covered by Computer Science. We will talk about Ecosystems, EMV spectrum and Rainbows using these analogies to state the necessity of integrating various fields of human endeveaours regarding the present day spread of, primarily, digital electronics. Though the basic components are in the very field of Electronics, digital technology, and therefore actually Computer Science, covers a huge area from interconnecting those electronic components into active units, up to enabling free telephone and videophone conversations through free wireless connection points.
In nature, an ecosystem is composed of living and nonliving entities that are connected and work together. Natural ecosystems are stable, as there is a homeostatic loop system between all their components (as said, living and nonliving). The second and even more important component of ecosystem stability is the self-organization, and between it and its wider environment. However, the area of usage of digital electronics and internetworking of all kinds of things grows stochastically, without regard to global inter-compatibility, and even less with regard to possible unknown unwanted consequences on our lives and our environment.
As already said, it is the role and responsibility of Computer Science to propose, test and apply a consistent cybernetic system which would enable stable development of the future Global Services Ecosystem, representing the Smart Service

System, which would integrate Human Knowledge and Intelligence and Computer Stubbornness and Exactitude, for the benefit of the human civilization and all individuals, by enabling but not forcing, giving more liberty and freedom and not less, to be a fascinating and powerful tool in our hands, but not by forcing itself onto us as a master.
Let us call this ecosystem The Rainbow Computing Ecosystem.

II. DISTRIBUTED SYSTEM EVOLUTION

Modern day computing paradigms foster for a huge community of involved participants from almost the whole spectrum of human endeavour. For computing and data processing there are; individual computers, their clusters, scientific Grids, and, finally, the Clouds/Fogs/Dews. For pure data communication there is the Internet, and for the Human-understandable Information Communication is the World Wide Web. The stunning development of actually extremely powerful hand-held mobile devices connected to the Internet enabled the "lowering" of certain parts of Clouds into the so called 'thin clients', and led to the development of the Fog Computing paradigm as well as to the ideas of Internet of Things (IoT) and the wish towards the Internet of Everything (IoE) what we propose to cowering by new Dew Computing paradigm. The tendency toward Dew computing is force by dynamic development of mobile computing, the decreasing availability and cost of computers and the huge number of networked components, devices and sensors in the networked environment. Dew computing is a distributed service technology as a ground part of architecture in cloud/fog/dew service which client data/info is processed at the periphery of the network, of source/process close as possible.

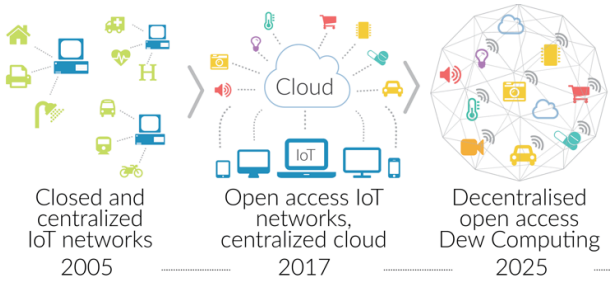


Fig 1 Distributed operation network evolution

Dew computing is a new computing paradigm appeared to fulfill applications at the edge of network in widely acceptance of cloud computing vertical hierarchy. Dew computing concerns the distribution of workloads between Cloud/Fog servers and local computers, and its focus is the application software organization of local computers. The goal of Dew computing is to fully realize the potentials of local computers in Cloud -Fog-Dew symbiosis [1].

Fig 2 presenting a complex service ecosystem of compatible federated components of Cloud – Fog – Dew Computing layers that all work together to enable a seamless system of global *Smart Service System* (SSS), proposed name *Rainbow*.

The Rainbow global service ecosystem will offer new applications and respond to changing business needs and support new business models, it will offer new possibilities of service processes development and information usage for a very broad user base, it will enable proper maintenance of essential natural and human-generated ecosystems, and enable huge savings and optimization in many areas of living and effort. Well integrated traffic systems, well cared for plants, efficient usage of energy, higher health level of general population, disease prevention, catastrophe warning/prevention, faster essential services,... all can be achieved by proper architectural means inside a Rainbow Service Ecosystem[2,3].

However the most significant amount of information processing all around us is done on the lowest possible computing level, outright connected to the physical environment and mostly directly controlling our human immediate surroundings. These "invisible", "embedded" information processing devices we find in everything from our car's motor, over air-conditioners, wending machines up to traffic-controls and wood-burning stoves, and ubiquitously all over the industry. These devices, which are neither at the Cloud/Fog edge, nor even at the mobile edge, but rather at the physical edge of computing are the basis of the Dew-Computing Paradigm.

The merits of including those "dew" devices into the Cloud - Fog - Dew hierarchy are huge, for individuals, the public and industrial sectors, the scientific community and the commercial sector, by bettering the physical and communicational, as well as the intellectual, immediate human environment.

III. WHY RAINBOW?

As the Sun by seemingly white light shines on Earth and gives it life, so the idea of *Rainbow* global distributed service is to incorporate all fields and segments of Computer Science into a unified entity, which will bring advancement and betterment of the quality of life of all people and our Planet.

As different frequencies of the Sunlight give different colours (spectrum), so different segments of Computer Science have important roles for the whole to function harmoniously. So let us borrow the analogy of Rainbow and its Colours for specific fields of Computer Science.

It is important to note here, that, in accordance with what was said before, in *Rainbow Computing* we necessarily include both the “technical” and the “philosophical” aspect, or, in other words, both the Machines and the Humans.

In the following two sections we will describe the overall Ecosystem from two major aspects: an integrated living social (human and machine) system, which we will visualize using the analogy of a Rainbow, i.e. sunlight Spectrum, and an architectural, more technically oriented, viewpoint.

The Colours of Rainbow

Red – *Basic Hardware*.

The general machine architecture layer, including hardware and associated machine code programming principles, partly also firmware and operating systems.

Present day computing hardware is mostly based on principles and architectures developed in the early days of computing, when many obstacles were to be overcome to get a viable computing machine. Nowadays it is essential to rethink much of the present day hardware platforms, as many possible avenues of computer architecture have not yet been experimented with, and the present day independent multicore serial processors approach, specifically due to a complete lack of proper human-oriented multi-processor/multi-computer programming principles and languages, is in many cases very inefficient.

Orange – *The Creativity Pool*.

Collection of creative ideas, with specific attention on novel approaches to teaching and education.

Many a problem, first regarded as separate and nonsolvable, suddenly got solved through some solution in a completely different area. Only if people have developed Creativity it is possible to have such insights, and Creativity is the main driving force of our civilization. But for proper and safe future the Creativity shall be primarily oriented towards high level integrated visions and properly established "why" and "do we really want/need/wish it", and only consequently the "hows". Education has also to be oriented towards the development of Creativity.

Yellow – *The Appropriateness Filter*.

Weighting of the level of concrete contribution of some advancement.

An idea, coming from the Orange layer must pass through the Appropriateness Filter, to test if its usage really contributes to concrete betterment of a specific field, or not, or could it in a wider context even be counterproductive or dangerous. Some ideas can be fanciful, exciting, beautiful, but it is not necessarily that they contribute in a positive sense to the civilisation changes, they may even be counterproductive in societal or environmental sense, and leading to worse general human living conditions. Generally, it seems that this area of Computer Science is well underdeveloped, and rarely the philosophical and primarily ethical wider consequences of introduction and interconnection of different new ideas and technologies are seriously considered, as they should be, at least scientifically, and consequently by recommendations and standards, in the most serious cases even by laws.

Green – *Environment and Health*.

Computer Science in the area of care for the general environment, environment control and maintenance and improvement of population health and general wellbeing.

This area involves health oriented devices and small gadgets, environmental sensors (and effectors), for example sensors in waterways reacting on specific unwanted chemicals, rain-composure sensors (e.g. rain acidity), light, humidity, temperature etc. sensors for closed environmental control, etc.
On the architectural plan, this area would be primarily covered by Dew Computing, due to the fact that most of these devices directly sense and influence the physical state of human and environmental well-being.

Blue – *Communication*.

This is the area of all kinds of communication and communication networks, including, naturally, the communication between computers and humans and vice versa.
Although generally it is that Computer Science is closely related to Cybernetics and Information Science, modern day development is aimed primarily towards Data. Data is, naturally, not Information, as it lacks the context/meanining, or, if you prefer, the Meta-Data. Present day communication both between humans and computers and between computers themselves lacks generic compatibility on the level of *what is exchanged*, as well as *what is to be done*.

Indigo – *Cooperation and Ethics*.

This area covers high level services based on compex communication interactions, the collection of huge quantities of Information (not Data, as is presently in vogue!) and the usage of that Information storage and processing.
An important aspect of future Computer Science development in this area will be the "pruning" of redundancy, and definition of long term safe redundancy levels of information important to our civilisation.

Violet – *Interference, Optics, Quanta*.

Protection and Expansion.

Our civilization is every day more and more, in some areas even already completely, dependent on computers and networks. We live in a world where even shorter disruptions of energy or information distribution systems can lead towards huge problems, from industry and traffic down to simple life at home. However, much of our electronic equipment is very suspect able to failure due to electric or electromagnetic discharges.
This failure suspectability, and its prevention is an extremely important area of future research and development due to our overall dependence on electrical systems, computer systems, wireless and satelite communications.

IV. THE ARCHITECTURE OF RAINBOW SERVICE

Architecturally speaking, Computer Science has already developed necessary defined notions and is intensively exploring necessary principles and solutions for a generic layered approach which can be developed into a firm fundament of the Rainbow Computing Ecosystem, as explained metaphorically through the above cited Colours of a Rainbow Spectrum.

In the area of Basic Hardware, those are presently primarily Clusters, Grids and GPU. However, a huge lack of a simple and consistent approach towards programming parallel distributed systems has a consequence of necessitating huge human efforts for attaining wished results. A generic ontology, as proposed for the Rainbow Ecosystem, with an appropriate high level human oriented language based on this ontology, with well defined grammar and semantics, and with a human approachable large dictionary (amount of recognized defined words) would allow proper integration of a myriad of generically heterogeneous devices, human users and the natural environment.

As basic architectural components presently we have the Cloud, Fog and Dew Computing layers. A *Rainbow* is a complex service ecosystem of interdependent components of Cloud – Fog – Dew Computing layers that all work together to announce the possibility of global services. As already mentioned, in nature an ecosystem is composed of living and nonliving entities that are interactively interconnected and work together to perform as a stable homeostatic and selforganising cybernetic system. The Rainbow service Ecosystem consists of hardware and software platforms/infrastructures/serveware[1], as well as service customers, engineers, consultants, integrators, providers and users, together with the human and natural environment. To achieve the homeostatic balance and selforganisation between those extremely differing components, consisting of three major groups – human made machines, natural humans and the planet Earth's nature (and with space exploration and satellite overcrowding even part of the outer space), each of them with their hugely different own specificities inside each group, it is essential to define a consistent system of future development towards the computing/services infrastructure being a simple and consistent, non-obtrusive and benevolent helper for human-oriented technical civilisation development.

Now time Cloud-Fog-Dew Computing defines important principles, but is usually thought of in terms of three broad service areas -- infrastructure-as-a-service (IaaS), platform-as-a-service (PaaS) and software-as-a-service (SaaS). However, these are not integrated as a unified "Smart Services System", and even less as a "Global Services Ecosystem". Naturally, the huge effort put into the development of these basic components promises a solid base for the high level integration provided by the Rainbow Service Ecosystem.

1 To enable the proper integration of all components "serveware", a service operational middleware, will have to be developed.

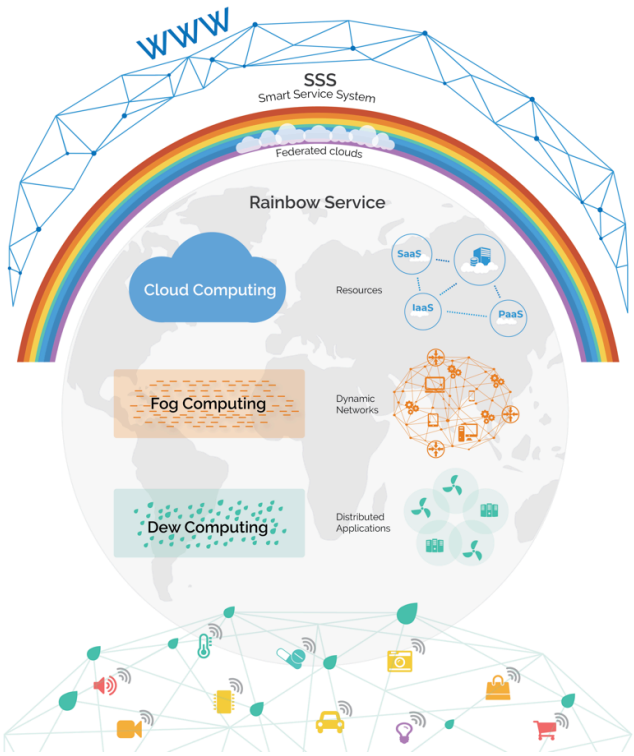


Figure 2 . Global Distributed Service Hierarchy

All three architectural machine levels – the Cloud, the Fog and the Dew have to grow into a compatible and inter-understandable Information System, integrated through ergonomics and linguistics with the other two essential components of our civilisation: the human world, society, production, consumption, politics, economy, and the natural world, wellbeing of all other inhabitants of our Planet, and non-interference into freedom of living and expression, both human and non-human (i.e. animals, plants, the Planet etc.).

Even today the Cloud services are really more complex then their generic descriptions, and the description of Cloud Computing also needs to include the vast array of service providers, service federations, harmonizations and orchestrations, i.e. the human component.

The Fog system, based primarily on an almost uncountable amount of individual programmable devices, has to provide seamless integration of both Dew Computing layer devices, Cloud Computing and individual Humans. Hierarchically, most of the ergonomics, language usage, information filtering and distribution shall be done in this layer, as it is the prime Rainbow Ecosystem layer connected directly to the Humans, in constant communication with them.

The Dew Computing components are, rather than communicating with Humans, actually the most sensitive area of Computing, as they are directly communicating, by sensing and effecting, with the physical and natural environment, often being able to directly change certain living conditions. The enormous vastness of very different sensing and effecting conditions and principles will necessarily generate an overwhelming amount of individual data. If we do not introduce a consistent ontology of meta-data and a linguistical framework of coordination

programming, filtering and sending/receiving, that is, if we do not start to use Information communication, providing each set of Data with appropriate Context, it will be impossible to properly integrate the Dew, Fog and Cloud Computing layers in a machine basis of the Rainbow Service Ecosystem.

Generally the full Ecosystem architecturally consists of five basic layers: Nature-Dew-Human-Fog-Cloud.

To enable the proper integration of all components by "serveware", a service operational middleware, will have to be developed. This serveware would be the prime information access, filtering and distribution component, and would include the full Rainbow Global Services Ecosystem ontology, as well as necessary linguistic elements (language) for human-computer and computer-computer communication (which shall be compatible, i.e. understandable by both humans and computers), and by which Information can be retrieved, processed and used. This may be achieved by selforganising autonomous service, which would organise the upward and downward information and request flow from Nature to Dew to Human to Fog to Cloud and vice versa.

In the previous section of this article a much more thorough description of scientific and human efforts, and the principles of proper integration of the human and natural world was done on a basis of analogies with Colours of a Rainbow. It is essential to mention that, similar to the Cloud-Fog-Dew layerisation of the machine aspect of a Global Services Ecosystem, or, as we call it poetically the Rainbow Ecosystem, it is necessary to do proper layerisation and define efficient interaction and intercommunication principles for the human components of the emerging Global Ecosystem.

Generally speaking we could enlist several layers of human-computer interaction levels: Users (simple or power), Scientists (innovators, researchers, students...), Infrastructuralists (networking, clustering...), Applicationists, Producers (in any production area), Companies (which necessary follow Economic trends), Societies (usage in Politics, everyday life, well-being etc.). These layers, though it may not be obvious from the first, are an essential field area of Computer Science, as the development of the future computing infrastructure has to be driven by and has to be controlled by these (broadly stated) "layers" of human society.

Therefore it is necessary to have an architectural system, which includes individual layer architectures (models) for all machine and human Computer Science fields, and includes an overall architecture, defining a viable selforganising cybernetic eco-system of the future human civilization.

V. IEEE DEWCOM STC COLLABORATION MODEL

IEEE Computer Society Dew Computing Special Technical Community (DewCom STC) under coordination of Y. Wang and K. Skala represent an open community and forum for researchers, professionals, and students in the area of Dew Computing and related distributed computing/service topics.

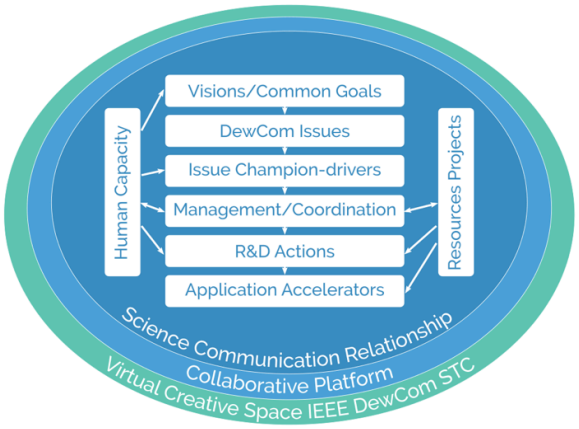


Figure 3 STC Collaboration model

The development of the Dew Computing paradigm as well as the vision of establishing a global smart service system (SSS) will be considering, developing, implementing through the IEEE DewCom Special Technical Community. For now STC has 43 members from 14 countries, presenting in fig.4. These virtual scientific research community establishing a collaborative platform for the realization of the *Dew Computing* paradigm and possibly *Rainbow* vision.

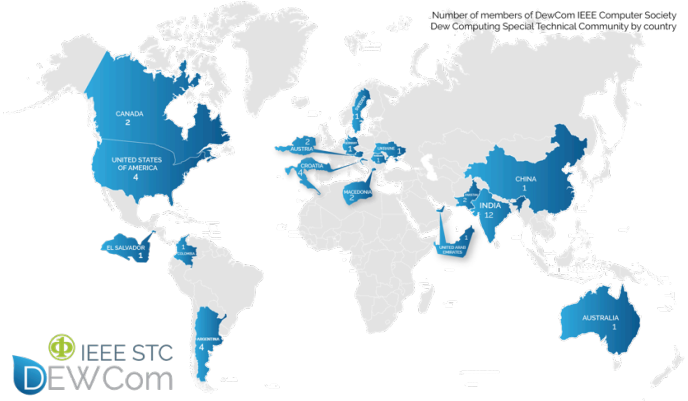


Fig 4 Membership distribution map

The Dew Computing development in organized manner intends, through Research, Innovation and Development, to ***explore*** the realm of possibilities of Dew-Computing, ***solve*** the basic problems of integration of the "dew" level with the higher level Dew-Fog-Cloud hierarchy, with special attention to the necessity of information (and not data) processing and communication, and ***demonstrate*** the

viability and high effectiveness of the developed Architecture in several areas of human endeavor through real life implementations. Finally, the IEEE STC collaborative action will ***define*** and, in cooperation with standardization/dictionary bodies, try to ***standardize*** the basics necessary for the seamless integration of the emerged Information Processing Architecture into the Dew, Fog and Cloud Paradigms, as a way towards the abovementioned civilization goals. Our intention is to work together as an virtual Research and development group to create, organize, share, and collaborate on projects and articles and develop our effective IEEE DewCom STC collaboration model.

VI. CONCLUSION

A robust *Rainbow Ecosystem* will offer new life-business applications and respond to changing civilization processes, business models, it will offer new possibilities of knowledge development and information usage for a very broad user base, it will enable proper maintenance of essential natural and human-generated ecosystems, and enable huge savings in many areas of effort. Well integrated traffic systems, well cared for plants, efficient usage of energy, higher health level of general population, disease prevention, catastrophe warning/prevention, faster essential services, ease of knowledge learning and information collecting, higher level of creativity education... all can be achieved by proper architectural means inside a *Rainbow Service Ecosystem*.

ACKNOWLEDGMENT

This research has been supported by the Ministry of Science and Education of the Republic of Croatia under the grant 533-19-15-0007 (Centre of Research Excellence for Data Science and Cooperative Systems) and IEEE Computer Society Dew Computing Special Technical Community (IEEE DewCom STC).

REFERENCES

[1] Skala, K., Davidović, D., Afgan, E., Sović, I., Šojat, Z.: Scalable distributed computing hierarchy: cloud, fog and dew computing. Open Journal of Cloud Computing (OJCC), 2 (1). pp. 16-24. ISSN 2199-1987, 2016

[2] Y. Wang, "Cloud-dew architecture," International Journal of Cloud Computing, vol. 4, no. 3, pp. 199–210, 2015.

[3] Pooja Kukreja, Deepti Sharma: A Detail Review on Cloud, Fog and Dew Computing, International Journal of Science, Engineering and Technology Research (IJSETR), Volume 5, Issue 5, 1412, 2016

[4] . Afgan, P. Bangalore, K. Skala: Application Information Services for Distributed Computing Environments, Journal of Future Generation Compute Systems, Volume 27, Issue, 2, p. 173-181, 2011.

Vehicular Data Analytics Dew Computing

Parimala Thulasiraman[1], Ruppa K. Thulasiram[1], Ying Ying Liu[2]

***Abstract*— Recent development of Intelligent Transportation System (ITS) offer several technologies. In centralized ITS solutions, such as Google map, crowd sourcing app Waze, Toyotas Entune dynamic routing, on board smart phone devices or vehicles send periodic local traffic data to a central server or cloud, and receive periodic traffic statistics on their routes of interest. The decentralized ITS solutions use Vehicular Ad Hoc Networks (VANET) to collect traffic information through vehicle-to-infrastructure (V2I) communication and vehicle-to-vehicle (V2V) communication. Cloud is a bottleneck in areas where Internet access may be limited. In this paper, we propose to develop a traffic aware routing algorithm on dew computers.**

I. INTRODUCTION

Real-word applications are complex networks, which can be represented abstractly as graphs. System entities in these scientific problems are the graph nodes and inter-relationship between these entities are the graph edges. The focus of this research is on one complex network, transportation networks.

Finding the optimal travel path from a source to destination on a given road map is called the vehicle routing problem. The road map is usually represented as a graph.The entities in transportation networks are landmarks, junctions or intersections and the links are the roads or lanes between these entities. In classical algorithms such as Dijkstras algorithm [1] and A* algorithm [2], the vehicle routing problem is solved by finding the shortest path on the road map with the weight on an edge representing the actual geometric distance between two junctions. These classical routing algorithms are static and do not consider the dynamic traffic information such as congestion, accidents and road closure. As vehicle traffic congestion becomes alarming severe in modern metropolitan areas, traffic-aware vehicle routing is one of the most important problems in improving quality of life and building smart cities with higher productivity, less air pollution and less fuel consumption. Recent development in Intelligent Transportation System (ITS)[3] offer several technologies.

In centralized ITS solutions, such as Google map, crowd sourcing app Waze, Toyotas Entune dynamic routing, on board smart phone devices or vehicles send periodic local traffic data to a central server or Cloud, and receive periodic traffic statistics on their routes of interest. The centralized ITS solutions are simple to implement and generally have good optimality due to the availability of global traffic information and powerful central computing power. However, such solutions require constant cellular connectivity of the smart phones or vehicles, and have a single point of failure on the centralized computing platform.

The decentralized ITS solutions use Vehicular Ad hoc Networks (VANET) to collect traffic information through vehicle-to-infrastructure (V2I) communication and vehicle-to-vehicle (V2V) communication. VANETs use IEEE 802.11p standard, an amendment to the IEEE 802.11 (Wi-Fi) standard, for dedicated short range communication (DSRC). In V2I communication, vehicles exchange information about specific road segment with nearby Road Side Units (RSU) via continuous wireless communication such as Wi-Fi hotspots or long/wide range wireless technologies. In V2V communication, vehicles are connected to nearby vehicles using short range wireless technologies. Compared to the centralized ITS solutions, VANETs are highly distributive and have lower cost for cellular bandwidth usage. Compared to cellular communication systems that only exist in some newer vehicles, embedded V2I/V2V communication systems are more prevalent in existing and new vehicles. Effective V2V communication is also much more economical than V2I communication, which requires the installation of RSUs and therefore investment in city infrastructure.

There are advantages and disadvantages of both V2I and V2V communication. The technology for V2I communication is well developed compared to V2V communication. However, V2V communication is a decentralized approach that can collect information in real time during a vehicle's movement on the road. The lack of technology to implement a decentralized environment is the main bottleneck in V2V communication. The proposed solution is to develop a traffic aware routing algorithm using dew computers.

II. RELATED WORK

There are two main challenges for dynamic real-time traffic-aware vehicle routing using V2V communication: 1) how to collect real-time traffic data, and 2) how to dynamically route a vehicle based on real-time traffic data as the vehicle travels on the road.

Collection of real-time traffic data requires efficient routing of data packet from vehicles to vehicles. In V2V communication, the connectivity between two vehicles is called a link. Due to the highly dynamic nature of vehicle mobility, and complex road condition and building blockage, VANETs are characterized by highly dynamic topologies with frequent link breakages, network fragmentation, and a high number of packet collisions and interferences. Therefore, studies of VANET protocols [4], [5], [6] have mainly focused on evaluating Quality of Service in delivery of arbitrary data packets

[1] Parimala Thulasiraman and Ruppa K. Thulasiram are with with Faculty of Science, Computer Science, University of Manitoba, Winnipeg, Canada, thulasir@cs.umanitoba.ca, tulsi@cs.umanitoba.ca [2] Ying Ying Liu is a PhD student in the Department of Computer Science, umliu369@myumanitoba.ca

from a source vehicle to a destination vehicle. However, the collection of traffic data requires generally involves multiple sources (the vehicles on the route of interest sending traffic data) and one destination (the vehicle requesting traffic data), which adds to number of hops and latency. Giuseppe, et al.[7] propose four V2V protocols for traffic congestion discovery along routes of interest through beacon messages.

After the collection of real-time traffic data, the next challenge is how to incorporate it into dynamic vehicle routing. In addition to improvements in traditional algorithms [8], [9], stochastic algorithms mimicking the routing of social animals in the dynamic nature have attracted much attention due to their proven efficiency and similarity to the dynamic vehicle routing problem. One popular algorithm is Ant Colony Optimization (ACO) [10], an iterative and evolving optimization heuristic. In nature, ants explore routes from nest to food source and deposit a chemical substance called pheromone, which attracts other ants to follow the same route. Pheromones evaporate over time. Eventually the longer paths lose pheromone concentration and all ants travel on the shortest path. For dynamic routing, Zhe et al. [11] develop a variant of ACO algorithm that uses stench pheromone to redirect ants to the second best route if the best route becomes too crowded. The authors incorporate traffic to the cost of each road segment as the total travel time on the segment. Jos Capela, et al [12] propose a hybrid algorithm of the Dijkstras algorithm and inverted ACO for traffic routing. These algorithms are centralized solutions that require global knowledge of the dynamic road network. In [13], the authors proposes efficient GSR, an improvement of the geographical source routing (GSR) protocol[6] using small controlled packets called ants to communicate traffic information and Dijkstra to recompute routes. This paper presents a methodology that incorporates traffic information in message communication routing for VANET, rather than actual vehicle routing. A communication route with optimum network connectivity is usually a road route with more traffic. In addition, the re-computation using Dijkstra's algorithm is expensive. In [15] we propose a local aware hybrid routing algorithm. In this algorithm, the VANET is divided into zones [14]. Each zone proactively determines the routing within its zone and reactively finds the routing paths within zones. The algorithm is robust to link failures and is scalable.

III. Proposed Solution

In this section we propose a solution that avoids using the Cloud or personal computers for data processing. The computations will be done on a dew computer. Each vehicle, now-a-days is equipped with lots of technological devices for performing many different functions such as heated seats, backup cameras, cruise control, key less entry, navigation, smart phone integration, automatic emergency braking and so on. In the next five years, we predict that chips with multiple processors will be installed on vehicles as they are on smart phones.

Drivers use GPS to navigate their route. The path finding algorithm is currently executed on Cloud and the information is sent directly to the vehicle. Sensors on road side units collect the necessary data to make accurate prediction of the route. The traditional path finding algorithms [1], [2] find only one path. The problem that may occur is the disruption of Internet services which will hinder in finding the route through Cloud. In the proposed solution, the routing algorithm finds multiple paths. We assume that there is a small and simple dew computer installed on each vehicle. These days graphic processors are cheap. Each dew computer will have a graphic processor installed to perform the computations.

The multiple path finding algorithm that we propose to use is the ant colony optimization algorithm. As per our previous work [15], we divide VANET into zones.The vehicles within the zone communicate with each other through short range communication medium. The vehicles exchange information about traffic conditions on the road. Using this information, the ant colony optimization algorithm is run on the road network.

Ant Colony Optimization is inspired by the foraging behaviour of ants searching for food. In nature, ants leave pheromone trails for the other members of the colony to follow their routes from nest to food source. The pheromones evaporate over time. As more ants travel a given path, the intensity of the pheromone increases. This leads to a better route and a shorter route.

The algorithm works in an iterative fashion. Initially, all the routes between source and destination are initialized with the same amount of artificial pheromone. In each iteration, two steps are performed: solution construction and pheromone update. In the phase of solution construction, virtual ants are scattered on the roads. Each ant follows a stochastic rule to choose the next intersection to visit, using a probability function that favors shorter tour and strong pheromone level. The phase ends when each ant completes finding the paths from source to destination. In the phase of pheromone update, pheromone level on all the routes evaporates (decreases) followed by an increment on the shortest tour in the iteration. The pheromone serves as a global memory, or an exploitation, to reinforce the local optimal. The stochastic rule allows ants to find multiple different solutions on several iterations and gives the algorithm enough exploration to find possibly better solutions.

The data collected through V2V and V2I communication mechanisms will serve as a basis for the routing algorithm. The ant colony optimization algorithm is highly parallelizable [16]. The algorithm can be easily parallelized on a graphic processor [17]. In our proposed scheme we will implement the ant colony optimization algorithm on the given road network on a vehicle's dew computer. The computer comprises of graphic processing units that will parallelize the algorithm and provide efficient routes in real time.

IV. Conclusion

The focus of this research is on intelligent transportation system. Predicting a route in the event of traffic congestion or other events on the road is challenging. Currently, Cloud

infrastructure is used to perform the routing computations. In this research, we propose to remove the Cloud infrastructure and perform the routing algorithm on dew computers installed on vehicles. We propose to develop a multi-path finding algorithm using a meta-heuristic inspired by real ants in nature.

ACKNOWLEDGMENT

The first authors thanks Research Manitoba for their funding support in conducting this research. All authors thank NSERC for their partial funding support.

REFERENCES

[1] E.W. Dijkstra, A note on two problems in connexion with graphs, Numerical Mathematics, 1, 1959, 269271.

[2] P.E. Hart, N.J. Nilsson and B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Transactions on Systems Science and Cybernetics. 4, 1968, 100107.

[3] G. Dimitrakopoulos and P. Demestichas, Intelligent transportation systems. IEEE Vehicular Technology Magazine, 5, 2010, 7784.

[4] C.E. Perkins and E.M. Royer, Ad hoc on demand distance vector (AODV) Routing, Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, 1999.

[5] B. Karp and H.T. Kung, GPSR: greedy perimeter stateless routing for wireless networks,MobiCom, 2000.

[6] C. Lochert, H. Fuler, H. Hartenstein, D. Hermann, J. Tian, and M. Mauve, A Routing strategy for vehicular ad hoc networks in city environments, Proceedings of the IEEE Intelligent Vehicles Symposium, Columbus, OH, June 2003.

[7] Giuseppe Martuscelli, Azzedine Boukerche, Luca Foschini and Paolo Bellavista, V2V protocols for traffic congestion discovery along routes of interest in VANETs: a quantitative study, Wireless Communications and Mobile Computing 16.17, 2016, 2907-2923.

[8] D.E. Kaufman DE and R.L. Smith, Fastest paths in time-dependent networks for intelligent vehicle-highway systems application, Journal of Intelligent Transportation Systems, 1(1), 1993, 111.

[9] L. Fu, An adaptive routing algorithm for in-vehicle route guidance systems with real-time information, Transportation Research B: Methodology, 35(8), 2001, 749765.

[10] Marco Dorigo and Gianni Di Caro, Ant colony optimization: a new meta-heuristic, Proceedings of the congress on evolutionary computation, Vol. 2. 1999.

[11] Zhe Cong, Bart De Schutter, and Robert Babuka, Ant colony routing algorithm for freeway networks, Transportation Research Part C: Emerging Technologies 37, 2013, 1-19.

[12] Dias, Jos Capela, et al, An inverted ant colony optimization approach to traffic, Engineering Applications of Artificial Intelligence 36, 2014, 122-133

[13] Forough Goudarzi, Hamid Asgari, and Hamed S. Al-Raweshidy, Traffic-Aware VANET Routing for City EnvironmentsA Protocol Based on Ant Colony Optimization." IEEE Systems Journal, 2018.

[14] Jianping Wang, Eseosa Osagie, Parimala Thulasiraman, and Ruppa K. Thulasiram. HOPNET: A hybrid ant colony optimization routing algorithm for mobile ad hoc network. Ad Hoc Networks, 7(4):690705, 2009.

[15] Himani Rana, Parimala Thulasiraman, and Ruppa K. Thulasiram, MAZACORNET: Mobility aware zone based ant colony optimization routing for VANET, IEEE Congress on Evolutionary Computation, Cancun, Mexico, 2013.

[16] Ziyue Wang, Ying Ying Liu, Parimala Thulasiraman and Ruppa K. Thulasiram, Ant Brood Clustering on Intel Xeon Multi-core: Challenges and Strategies, IEEE Symposium Series on Computational Intelligence (SSCI), Banglore, India, 2018.

[17] Audrey Delvacq, Pierre Delisle, Marc Gravel and Michal Krajecki, Parallel ant colony optimization on graphics processing units, Journal of Parallel and Distributed Computing, 73(1), 2013, 52-61.

Dewblock: A Blockchain System Based on Dew Computing

Yingwei Wang
School of Mathematical and Computational Sciences
University of Prince Edward Island
Charlottetown, Canada
Email: ywang@upei.ca

***Abstract*—The blockchain technology enabled cryptocurrencies and a lot of other applications that trust is needed among different entities. Because every blockchain client needs to keep huge amount of blockchain data, some personal computers and mobile devices cannot be used to run blockchain clients. To make things worse, the size of blockchain data is always increasing. In this paper, a new kind of blockchain system, Dewblock, is introduced. In this system, a blockchain client does not need to keep the blockchain data and it also has the features of a blockchain full node. Dewblock was developed based on dew computing principles and architecture.**

***Index Terms*—Blockchain; Dew computing; Cloud-dew architecture; Cloud services; Blockchain full client; Blockchain lightweight client.**

I. Introduction

Blockchain was first introduced with Bitcoin, a cryptocurrency, but blockchain is not limited to cryptocurrencies. It can be used in various occasions. Ginni Rometty, the CEO of IBM, once said: "Blockchain will do for transactions what the internet did for information" [1].

Blockchain has a feature that limits the range of its applications: each blockchain full client has to keep the whole blockchain starting from the genesis block; this blockchain gets longer and longer with the operation of the blockchain network. For this reason, a blockchain full client is not suitable to be deployed to personal computers and mobile devices.

It is desirable to find solutions to tackle the above problem so that the data size of a blockchain client can be reduced. Such kind of solutions are hard to find because this problem comes with the essential nature of blockchain. Blockchain data itself is the heart of the blockchain technology; the data size of a blockchain client is inherently big and inherently keeps increasing.

For some cryptocurrency systems, such as Bitcoin and Ethereum, their blockchains already have huge amount of data; a quite powerful computer is needed to run a full client. To make these cryptocurrency systems accessible by users, blockchain lightweight clients were developed and are widely used. These lightweight clients do not need to keep the whole blockchain data so that they can be deployed to personal computers and mobile devices. These lightweight clients include SPV (Simple Payment Verification) wallets, such as Electrum, Copay, and so on.

All the lightweight clients are not qualified as full clients. They do whatever the majority of mining power says. They rely on the support provided by full clients. These lightweight clients are necessary and they are playing important roles in the cryptocurrency systems. But the goal of this paper is not to find another lightweight client.

Blockchains can be used in wide range of areas. Various blockchain systems will be developed in the future for different kinds of transactions. We want to propose a generic blockchain client architecture so that these clients can be deployed to personal computers and mobile devices and these clients still have features of full nodes.

With such goals in mind, we would like to introduce a new blockchain system: Dewblock [2]. Dewblock's dew clients do not keep blockchain data so that their data size is very small; Dewblock's dew clients still have features of full nodes; Dewblock is based on dew computing principles and architecture [3][4].

The rest of the paper is organized as follows: Section II discusses the two models of blockchains and indicates that the Dewblock approach can only be applied to blockchains based on one model. The good news is that the model of a blockchain can be changed. Section III, Section IV, and Section V introduce the key concepts of Dewblock. Section VI introduces the Dewblock project and its resources. Finally, Section VII is devoted to conclusions.

II. State-keeping Models

The approach to control client data size that we are going to introduce in this paper cannot be applied to arbitrary kinds of blockchain systems. To determine each blockchain's suitability to our new approach, we discuss the state-keeping models of blockchains in this section.

Blockchains can be considered as state transition systems or state machines [5]. Different state-keeping models can be used. Two types of state-keeping models are popular in todays blockchain networks. The first model is the *unspent transaction output (UTXO) model*. The second one is the *account model*. For example, Bitcoin uses the UTXO model [6], and Ethereum uses the account model [5].

Being the first blockchain system, Bitcoin is operated using the UTXO model. In the UTXO model, each transaction spends output from prior transactions and generates new

outputs that can be spent by transactions in the future. All of the unspent transactions are kept in each full client. A user's wallet keeps track of a list of unspent transactions associated with all addresses owned by the user, and the balance of the wallet is calculated as the sum of those unspent transactions.

The account model, on the other hand, keeps track of the balance of each account as a global state. When a transaction is being verified, the balance of an account is checked to make sure it is larger than or equal to the spending amount of the transaction.

Each of these two models has its advantages and disadvantages. The features of these two models have been discussed in literature [7][8]. From our viewpoint, we believe that the difference between these two models is that they have different thinking logic or different philosophy: the UTXO model is history oriented; the account model is reality oriented.

As Satoshi Nakamoto mentioned in his historic paper [6]: "We define an electronic coin as a chain of digital signatures. Each owner transfers the coin to the next by digitally signing a hash of the previous transaction and the public key of the next owner and adding these to the end of the coin. A payee can verify the signatures to verify the chain of ownership." Using an analogy, if you want to verify a coin is true in our daily life, you have to go over all the transactions this coin went through: the transaction that person A gave the coin to you; the transaction that person B gave the coin to person A, and so on, until the coin was made in the mint.

This logic works, and might be meaningful in some sense, but it is in contrary to our daily practice. Using this model, all the transactions since the start of the blockchain should exist and ready for verification. Banks usually keep detailed transaction history for quite a long time, but no bank will keep all its transactions forever. As time goes by, the size of the transaction history gets bigger and bigger. This model is not sustainable; at least it is not sustainable for most clients in a blockchain network.

Using account is our daily practice in keeping records. People's money saved in bank accounts. Each student has a profile in his/her university and this profile is called an account. The key points of an account-based system is that the accounts reflect current state of the system and these accounts do not rely on the complete history of the system, whether the system is a bank, an organization, or a blockchain.

Our efforts to develop small-data-size blockchain clients shall be based on the account-model blockchains instead of the UTXO-model ones. This restriction does not limit this approach's application because UTXO-model blockchains can be converted into account-model blockchains.

The new blockchain system we are going to introduce, Dewblock, was developed from another blockchain system: Naivecoin [9]. Naivecoin uses UTXO model. To reach our goals, we have switched Naivecoin from the UTXO model to the account model. For convenience of discussion, we would like to give a name to the Naivecoin system that has been switched to account model: *Account-Naivecoin*.

The successful conversion of Naivecoin from the UTXO model to the account model shows that the new approach we have introduced in Dewblock can be applied to all blockchain systems, although a conversion might be needed. In the future, when new blockchain systems are designed for various kinds of transactions, account model shall be used if we want to adopt the Dewblock approach.

III. Dewblock Architecture: Cloud-dew Architecture

Dewblock is designed based on cloud-dew architecture [3]. A Dewblock package is composed of a cloud server and a dew server; the cloud server and the dew server talk to each other through a new type of message channel; the dew server operates in two different modes. These topics will be discussed in the following subsections.

A. Cloud Server and Dew Server

To introduce cloud-dew architecture to Account-Naivecoin blockchain system, we would use two copies of Account-Naivecoin client. Using the terminology of cloud-dew architecture, we call one Account-Naivecoin client *cloud server*, and call the other Account-Naivecoin client *dew server*. Here we need to clarify a few terms. The term *client* which appeared in blockchain client, full client, lightweight client, Account-Naivecoin client means a client of the blockchain network. The term *server* which appeared in cloud server and dew server means a server to a user. Thus a cloud server or a dew server could act as a blockchain client; a blockchain client could be considered as a cloud server or a dew server. In the rest of the paper, terms cloud server and dew server refer to software or program package; terms dew client and full client refer to this software's role in a blockchain network.

The names cloud server and dew server make sense because the cloud server usually be deployed to a public cloud service, such as Amazon Web Services or Google App Engine, or a private cloud service where the environment is configured to support such cloud servers, and the dew server usually be deployed to a personal computer or a mobile device.

The dew server is basically a copy of the cloud server but is not necessarily the exact copy of the cloud server. In this case, we would like to introduce an important and interesting difference between the cloud server and the dew server: The cloud server contains the blockchain but the dew server does not contain the blockchain. In such a system, the data size of the dew server would be quite small.

Before we go further from here, we should make one thing clear: Account-Naivecoin uses account model, but it does keep the blockchain. Can a dew server operate without a blockchain? In other words, is it possible for an Account-Naivecoin client without the blockchain to work with the rest of the Account-Naivecoin blockchain network?

The answers to the above questions are both positive. We may modify the dew server copy of the Account-Naivecoin so that it does not keep the blockchain but otherwise it still operates in the same way. In an Account-Naivecoin blockchain

network, if one client throws out the blockchain, it can still operate well in the network: it can make transactions; it can maintain the transaction pool; it can communicate with other clients; it can verify if a transaction is valid; it can even mine a new block. The only problem this client has is that when another client asks this client to provide the whole blockchain, this client cannot respond properly.

An Account-Naivecoin client without the blockchain can operate for most of the cryptocurrency functions, but it is not a blockchain full node. This client does not have enough strength to fight attacks. If only a few clients work this way, the whole blockchain network would still run properly; if many clients are not full nodes, the whole blockchain network would deteriorate, and the trust brought in through blockchain would be gone.

In Dewblock, a dew server is not a blockchain full node, but a cloud server is. The pair of a cloud server and a dew server can also serve as one single full node to the rest of the Dewblock network. In blockchain terminology, node and client were considered the same. In Dewblock, node and client are not always the same any more. A *blockchain client* is a program that a user installed in his/her computer or device to operate a blockchain network. A *blockchain node* is a logical unit that acts as one single identity in a blockchain network. A node may contain a cloud server and a dew server, but a client is always a dew server.

B. Message Channels

In an Account-Naivecoin network, only one kind of message channel exists: *inter-node channel*. When the cloud-dew architecture was introduced, another kind of communication channel was needed for cloud servers and dew servers to collaborate. The new kind of message channel between cloud servers and dew servers is called *cloud-dew channel*. WebSocket protocol was used to create such channels.

In an Account-Naivecoin network, five kinds of messages travel through the inter-node channels. In a Dewblock network, four more kinds of messages were added and they could travel through the inter-node channels and the new cloud-dew channels. Some messages can travel in both kinds of channels; some messages can only travel in one specific kind of channel. The details of the message mechanism can be found in the website http://www.dewblock.com. In the rest of this section, Section IV, and Section V we will explain the rationals related to the four kinds of newly-added message types.

Two new message types are introduced to transfer account information through cloud-dew channels:

- QUERY_ACCOUNTS
- RESPONSE_ACCOUNTS.

To make sure that the account in a dew server is consistent with the account in a cloud server, the dew server will periodically send its account information to the cloud server for verification. If discrepancy is found, the account information in the cloud server will be fetched to the dew server to replace the account information in the dew server.

C. Simple Node and Cloud-dew Node

In some situations, a dew server may want to operate as a full client for various reasons. Such option should be provided to users in case it is necessary. Thus a dew server can operate in one of the two modes: *dew mode* and *full mode*.

When a dew server is in dew mode, it behaves as described in Section III-A and Section III-B, and we call this dew server a *dew client*. The cloud server and the dew client constitute a single Dewblock node, and we call this node a *cloud-dew node*.

When a dew server is in full mode, it behaves the same with an full Account-Naivecoin client, and we call the dew server a *full client*. In this mode, the dew server itself constitutes a Dewblock full node, and we call this node a *simple node*. The cloud server is not involved in the operation of a simple node; it may be turned off, may be operated as a separate node, or may be even not deployed at all.

From here on, we may use terms *dew client* or *full client* to replace the term *dew server* whenever it is appropriate. With these terms, the mode of the dew server is indicated.

A dew server can change its mode: it can be switched from full mode to dew mode, or vice verse.

When a dew server is switched from local node to dew mode, it needs to establish the cloud-dew channel with the cloud server described in its configuration file, to establish its connections with other nodes as described in Section IV, and to discard the blockchain to become a small-data-size dew client.

When a dew server is switched from dew mode to full mode, it needs to obtain the blockchain from another place. One of the possibilities is to establish an inter-node channel with the cloud server to obtain the blockchain. The cloud-dew channel between the cloud server and the dew server shall be cut off. It also needs to re-establish its connections with other nodes according to its new role.

IV. Pair Connection Protocol

There are two types of nodes in a Dewblock network: simple nodes and cloud-dew nodes. Here we discuss the process to establish connections between nodes. First, proper connections can be described in the following:

- When two simple nodes are getting connected, one inter-node channel is needed to connect them.
- When one simple node and one cloud-dew node are getting connected, two inter-node channels are needed: one to connect the simple node to the dew client of the other node and one to connect the simple node to the cloud server of the other node.
- When two cloud-dew nodes are getting connected, two inter-node channels are needed: one to connect the two dew clients of the two nodes and one to connect the two cloud servers of the two nodes.

We make a few assumptions:

- Connections are initiated by dew clients or full clients.

- A client knows its own mode. If it is in dew mode, it knows the address of its cloud server through its configuration file.
- A client needs to know the address of another client to establish a connection.
- A client does not have to know the types of other clients (full clients or dew clients) , although it may get to know their types after exchanging messages.

We need to create rules so that connections between nodes can be properly established. For the convenience of discussion, we use an analogy to describe the above situation.

In a community, people need to get connected. We make the following assumptions:

- Every person is in a family. A family could have a gentleman and a lady or a single lady. A family cannot only have a single gentleman.
- Every family has a contact person. A gentleman or a single lady is the contact person. Connections could be initiated by any contact person to any other contact person. Somehow contact persons can find each other. Each contact person decides if a connection request will be accepted.
- A proper connection between two families can be described in the following: a gentleman is connected to a gentleman; a lady is connected to a lady; a gentleman is connected to a lady only when the lady is single.

We create the following rules so that proper connections among families can be established.

Gentleman's Rule:

- Whenever he is involved in a connection, actively or passively, he would make an introduction: "My name is Mr. Blah. It is my honor to introduce my wife Mrs. Blah to you."
- Whenever he receives such an introduction from another gentleman, he passes the introduction to his wife.

Lady's Rule:

- Whenever she receives an introduction from her husband or another gentleman, if she does not know the introduced lady yet, she would connect with that lady and tell her who introduced her.
- Whenever she receives a connection request and was told who introduced her, she will accept the request only if the introduction was from her husband.

The above rules can make sure all the ladies and gentleman are properly connected. We refer to these rules as *Pair Connection Protocol*.

To implement the Pair Connection Protocol in Dewblock, one more type of message was added. This message type is ALTERNATE_ADDRESS. This type of message can travel in both inter-node channels and cloud-dew channels.

Whenever a dew client initiates a connection or receives a connection request, it sends out an ALTERNATE_ADDRESS message to the other end of the connection. Whenever a dew client receives such a message, it passes this message to its cloud server.

Whenever a cloud server or a full client receives an ALTERNATE_ADDRESS message, it first checks if such a connection already exists; if not, it initiates a connection with the address specified in the message and sends this message to the receiver. When a cloud server receives a connection request and an ALTERNATE_ADDRESS message, it verifies that the message was originated from this cloud server's dew server before it accepts the connection.

V. Collaboration Mechanisms

Let us continue to use the family analogy introduced in Section IV to describe Dewblock's collaboration mechanisms. These descriptions further reveal the features of Dewblock, and also demonstrate the important role of family analogy in inspiring and explaining these mechanisms.

A. Integrity Keeping

If a dew client can perform all cryptocurrency operations, why do we need a cloud server? In other words, why do we need to operate a full node? The following analogy provides an explanation.

All families in a community need to maintain the community's justice and well being. They need not only to work for their own families, but also to vote for the community for various reasons. Every family should have a voter. To satisfy this requirement, every family designates the lady of the family as the voter. Whenever a vote is called, the gentlemen would ignore the call, but the ladies would vote. In a community with strict rules, if a family does not participate voting for a while, this family shall be excluded from the community.

In Dewblock, we have a similar situation. Each node has its responsibility to keep the integrity of the blockchain network. Each node needs not only to operate the node's own functions, but also to keep the whole blockchain and provide the whole blockchain to other nodes when needed. The cloud server or the full client of each node are designated to fulfill this responsibility. The dew client of each node would ignore the request to provide the whole blockchain. Strictly speaking, If a node does not fulfill its responsibility in keeping the integrity of the blockchain network for a while, this node might be disconnected from the network. This exclusion rule has not been implemented in Dewblock code yet and is on our future agenda.

B. Mining in Cloud

Block mining is an important activity in blockchains. How is mining performed in Dewblock? Let us check the similar situation in the family analogy first.

All families in a community have meals together in a shared fashion. It is an honour for a family to cook for the community. Gentlemen can cook, but ladies cook better. Single ladies know when to cook, but wives only cook when their husbands ask them to do so.

Let us go back to Dewblock. Dew clients can perform block mining. Because mining takes huge amount of computing power, it may seriously influence the normal operation of a

personal computer or a mobile device where the dew client is running. A better arrangement would be to ask the cloud server to mine a new block on behalf of the dew client. A new kind of message, MINING_REQUEST, was introduced. This kind of message travels through cloud-dew channels. When such a message is received, the cloud server tries to mine a new block; if successful, the new block will be added to the Dewblock network.

VI. Dewblock Project

Dewblock is a blockchain cryptocurrency system that was developed as a proof-of-concept system for the principles described in this paper. It can be modified to accommodate transactions on records other than cryptocurrencies.

Dewblock was developed on top of an open source project Naivecoin [9][10]. Naivecoin is a blockchain cryptocurrency system. It tries to show that the basic principles in a cryptocurrency can be implemented in a concise way. Naivecoin uses the UTXO model.

Dewblock was developed through the following major changes to Naivecoin:

- Naivecoin's underlying state-keeping model has been changed from the UTXO model to the account model. Such modified Naivecoin was referred to as Account-Naivecoin in this paper.
- Cloud-dew architecture was introduced. Dewblock contains two packages: *Dewblock-cloud-server* and *Dewblock-dew-server*. These two packages are modified versions of Account-Naivecoin.
- Dew servers can operate in two different modes: dew mode and full mode. When a dew server is in dew mode, it operates without the blockchain.
- A new kind of message channel, cloud-dew channel, was added. Four new types of messages were added.
- Pair Connection Protocol was implemented.
- Collaboration mechanisms, such as integrity keeping and mining in cloud, were implemented.
- Web commands were updated; configuration files were added.

The source code of Dewblock is stored in Github [11] as open source software under Apache License. The implementation details and operation instructions can be found in http://www.dewblock.com.

The most significant feature of Dewblock is that it provides a dew client; the dew client does not keep blockchain data; the dew client is part of a blockchain full node. Such dew clients can be introduced in various blockchain systems and deployed to personal computers and mobile devices.

VII. Conclusions

At the heart of the blockchain technology, a distributed secure ledge, a blockchain, is stored in every node of the network and trust can be established among unknown parties. By definition, blockchain data has to exist in every node and the data amount increases with time. Thus, the problem we are trying to tackle, blockchain clients' data size is too big and always increasing, is inherent to this technology. In the past, some approaches have been proposed to reduce the data size of blockchain clients, but these clients do not have the status of full nodes. Although these approaches provide convenience to users, they cannot be used in the backbone of blockchain networks.

Dewblock brings in a new approach that the data size of a client is reduced and the features of a full node are still kept. The key point is that the two concepts, blockchain client and blockchain node, are not the same any more in Dewblock. While a client is light-weighted and is conveniently operated in a personal computer or a mobile device, the client works with a remote cloud server to act as a full node.

This approach was inspired by dew computing principles. The architecture of Dewblock is the cloud-dew architecture. A dew client operates independently to perform blockchain activities; it also collaborates with the cloud server to maintain the integrity of the whole blockchain network. Two major features of dew computing, independence and collaboration, are demonstrated clearly in this application.

With Dewblock, each blockchain user needs to deploy a cloud server to a cloud service. From technical and economical viewpoint, the widely use of cloud services by individual users, including blockchain users, is feasible and affordable. Apparently, Dewblock, as a dew computing application, promotes the usage of cloud computing. This fact demonstrates the relationship between dew computing and cloud computing: cloud computing enabled dew computing; dew computing further promotes cloud computing; dew computing is the complementary piece of cloud computing [12].

References

[1] Sthuthie Murthy. (2018, May) "Blockchain will do for transactions what the internet did for information" - says IBM CEO. [Online]. Available: https://ambcrypto.com/blockchain-for-transactions-internet-for-information-ibm-ceo/

[2] Yingwei Wang. (2018, Sept.) Dewblock. [Online]. Available: http://www.dewblock.com/

[3] Y. Wang, "Cloud-dew architecture," *International Journal of Cloud Computing*, vol. 4, no. 3, pp. 199–210, 2015.

[4] Yingwei Wang, "Definition and categorization of dew computing," *Open Journal of Cloud Computing (OJCC)*, vol. 3, no. 1, pp. 1–7, 2016.

[5] Vitalik Buterin. (2013, Dec.) A next-generation smart contract and decentralized application platform. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper#ethereum

[6] Satoshi Nakamoto. (2009, May) Bitcoin: A peer-to-peer electronic cash system. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[7] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," *IEEE Access*, vol. 4, pp. 2202–2303, 2016.

[8] Brian Curran. (2018, Jul.) Comparing bitcoin & ethereum: UTXO vs account based transaction models. [Online]. Available: https://blockonomi.com/utxo-vs-account-based-transaction-models/

[9] lhartikk. (2017, Dec.) Naivecoin: a tutorial for building a cryptocurrency. [Online]. Available: https://lhartikk.github.io/

[10] ——. (2017, Dec.) Naivecoin. [Online]. Available: https://github.com/lhartikk/naivecoin

[11] Yingwei Wang. (2018, Aug.) Dewblock. [Online]. Available: https://github.com/yingweiwang/dewblock

[12] A. Rindos and Y. Wang, "Dew computing: The complementary piece of cloud computing," in *Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)(BDCloud-SocialCom-SustainCom), 2016 IEEE International Conferences on*. IEEE, 2016, pp. 15–20.

Keyword Index