An International ICT R&D Journal Sponsored by ZTE Corporation

ISSN 1673-5188
CN 34-1294/ TN
CODEN ZCTOAK

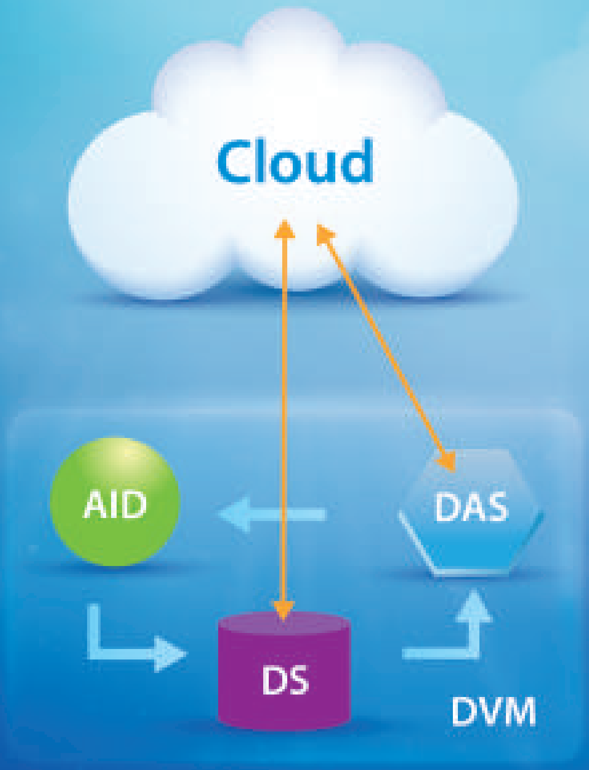
ZTE COMMUNICATIONS

ZTE中兴 ZTE COMMUNICATIONS

tech.zte.com.cn

October 2017, Vol. 15 No. 4

SPECIAL TOPIC:
Cloud Computing, Fog Computing, and Dew Computing



VOLUME 15 NUMBER 4 OCTOBER 2017

9 771673 518178 10

ZTE Communications Editorial Board

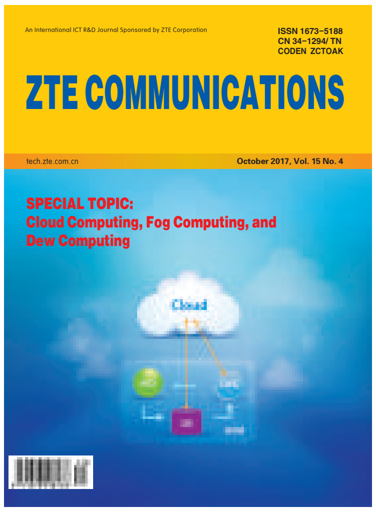
Chairman **ZHAO Houlin**: International Telecommunication Union (Switzerland)

Vice Chairmen **ZHAO Xianming**: ZTE Corporation (China) **XU Cheng-Zhong**: Wayne State University (USA)

Members (in Alphabetical Order):

CAO Jiannong	Hong Kong Polytechnic University (Hong Kong, China)
CHEN Chang Wen	University at Buffalo, The State University of New York (USA)
CHEN Jie	ZTE Corporation (China)
CHEN Shigang	University of Florida (USA)
CHEN Yan	Northwestern University (USA)
Connie Chang-Hasnain	University of California, Berkeley (USA)
CUI Shuguang	University of California, Davis (USA)
DONG Yingfei	University of Hawaii (USA)
GAO Wen	Peking University (China)
HWANG Jenq-Neng	University of Washington (USA)
LI Guifang	University of Central Florida (USA)
LUO Fa-Long	Element CXI (USA)
MA Jianhua	Hosei University (Japan)
PAN Yi	Georgia State University (USA)
REN Fuji	The University of Tokushima (Japan)
SONG Wenzhan	University of Georgia (USA)
SUN Huifang	Mitsubishi Electric Research Laboratories (USA)
SUN Zhili	University of Surrey (UK)
Victor C. M. Leung	The University of British Columbia (Canada)
WANG Xiaodong	Columbia University (USA)
WANG Zhengdao	Iowa State University (USA)
WU Keli	The Chinese University of Hong Kong (Hong Kong, China)
XU Cheng-Zhong	Wayne State University (USA)
YANG Kun	University of Essex (UK)
YUAN Jinhong	University of New South Wales (Australia)
ZENG Wenjun	Microsoft Research Asia (USA)
ZHANG Chengqi	University of Technology Sydney (Australia)
ZHANG Honggang	Zhejiang University (China)
ZHANG Yueping	Nanyang Technological University (Singapore)
ZHAO Houlin	International Telecommunication Union (Switzerland)
ZHAO Xianming	ZTE Corporation (China)
ZHOU Wanlei	Deakin University (Australia)
ZHUANG Weihua	University of Waterloo (Canada)

CONTENTS


An International ICT R&D Journal Sponsored by ZTE Corporation
ISSN 1673-5188
CN 34-1294/ TN
CODEN ZCTOAK
ZTE COMMUNICATIONS
tech.zte.com.cn
October 2017, Vol. 15 No. 4
SPECIAL TOPIC:
Cloud Computing, Fog Computing, and Dew Computing
Cloud

Special Topic: Cloud Computing, Fog Computing, and Dew Computing

Guest Editorial 01
PAN Yi and LUO Guangchun

A Transparent and User-Centric Approach to Unify Resource Management and Code Scheduling of Local, Edge, and Cloud 03
ZHOU Yuezhi, ZHANG Di, and ZHANG Yaoxue

An OS for Internet of Everything: Early Experience from A Smart Home Prototype 12
CAO Jie, XU Lanyu, Raef Abdallah, and SHI Weisong

HCOS: A Unified Model and Architecture for Cloud Operating System 23
CHEN Aiguo, WU Huaigu, TIAN Ling, and LUO Guangchun

Dew Computing and Transition of Internet Computing Paradigms 30
WANG Yingwei, Karolj Skala, Andy Rindos, Marjan Gusev, YANG Shuhui, and PAN Yi

Online Shuffling with Task Duplication in Cloud 38
ZANG Qimeng and GUO Song

Submission of a manuscript implies that the submitted work has not been published before (except as part of a thesis or lecture note or report or in the form of an abstract); that it is not under consideration for publication elsewhere; that its publication has been approved by all co-authors as well as by the authorities at the institute where the work has been carried out; that, if and when the manuscript is accepted for publication, the authors hand over the transferable copyrights of the accepted manuscript to *ZTE Communications*; and that the manuscript or parts thereof will not be published elsewhere in any language without the consent of the copyright holder. Copyrights include, without spatial or timely limitation, the mechanical, electronic and visual reproduction and distribution; electronic storage and retrieval; and all other forms of electronic publication or any other types of publication including all subsidiary rights.

Responsibility for content rests on authors of signed articles and not on the editorial board of *ZTE Communications* or its sponsors.

All rights reserved.

CONTENTS

ZTE COMMUNICATIONS
Vol. 15 No. 4 (Issue 59)
Quarterly
First English Issue Published in 2003

Supervised by:
Anhui Science and Technology Department

Sponsored by:
Anhui Science and Technology Information Research Institute and ZTE Corporation

Staff Members:
Editor-in-Chief: CHEN Jie
Executive Associate Editor-in-Chief: HUANG Xinming
Editor-in-Charge: ZHU Li
Editors: XU Ye, LU Dan, ZHAO Lu
Producer: YU Gang
Circulation Executive: WANG Pingping
Assistant: WANG Kun

Editorial Correspondence:
Add: 12F Kaixuan Building,
329 Jinzhai Road,
Hefei 230061, China
Tel: +86-551-65533356
Fax: +86-551-65850139
Email: magazine@zte.com.cn

Published and Circulated (Home and Abroad) by:
Editorial Office of *ZTE Communications*

Printed by:
Hefei Tiancai Color Printing Company

Publication Date:
October 25, 2017

Publication Licenses:

ISSN 1673-5188
CN 34-1294/ TN

Annual Subscription:
RMB 80

Technical Analysis of Network Plug-In Flannel for Containers 43
YANG Yong, DONG Xiugang, and DONG Zhenjiang

Review

Virtualization Technology in Cloud Computing Based Radio Access Networks: A Primer 47
ZHANG Xian and PENG Mugen

Introduction to *ZTE Communications*

ZTE Communications is a quarterly, peer-reviewed international technical journal (ISSN 1673-5188 and CODEN ZCTOAK) sponsored by ZTE Corporation, a major international provider of telecommunications, enterprise and consumer technology solutions for the Mobile Internet. The journal publishes original academic papers and research findings on the whole range of communications topics, including communications and information system design, optical fiber and electro-optical engineering, microwave technology, radio wave propagation, antenna engineering, electromagnetics, signal and image processing, and power engineering. The journal is designed to be an integrated forum for university academics and industry researchers from around the world. *ZTE Communications* was founded in 2003 and has a readership of 5500. The English version is distributed to universities, colleges, and research institutes in more than 140 countries. It is listed in Inspec, Cambridge Scientific Abstracts (CSA), Index of Copernicus (IC), Ulrich's Periodicals Directory, Abstract Journal, Chinese Journal Fulltext Databases, Wanfang Data — Digital Periodicals, and China Science and Technology Journal Database. Each issue of *ZTE Communications* is based around a Special Topic, and past issues have attracted contributions from leading international experts in their fields.

Guest Editorial

PAN Yi and LUO Guangchun

Cloud Computing, Fog Computing, and Dew Computing

▶ PAN Yi

Dr. PAN Yi is currently a Regents' Professor and Chair of Computer Science at Georgia State University, USA. He has served as an Associate Dean and Chair of Biology Department during 2013-2017 and Chair of Computer Science during 2006-2013. Dr. PAN joined Georgia State University in 2000, was promoted to full professor in 2004, named a Distinguished University Professor in 2013 and designated a Regents' Professor (the highest recognition given to a faculty member by the University System of Georgia) in 2015. Dr. PAN received his B.Eng. and M. Eng. degrees in computer engineering from Tsinghua University, China, in 1982 and 1984, respectively, and his Ph.D. degree in computer science from the University of Pittsburgh, USA, in 1991. His profile has been featured as a distinguished alumnus in both Tsinghua Alumni Newsletter and University of Pittsburgh CS Alumni Newsletter. Dr. PAN's research interests include parallel and cloud computing, wireless networks, and bioinformatics. Dr. PAN has published more than 330 papers including over 180 SCI journal papers and 80 IEEE/ACM Transactions papers. In addition, he has edited/authored 40 books. His work has been cited more than 8300 times in GoogleScholar. Dr. PAN has served as an editor-in-chief or editorial board member for 15 journals including 7 IEEE Transactions. He is the recipient of many awards including IEEE Transactions Best Paper Award, 4 other international conference or journal Best Paper Awards, 4 IBM Faculty Awards, 2 JSPS Senior Invitation Fellowships, IEEE BIBE Outstanding Achievement Award, NSF Research Opportunity Award, and AFOSR Summer Faculty Research Fellowship. He has organized many international conferences and delivered keynote speeches at over 50 international conferences around the world.

▶ LUO Guangchun

Dr. LUO Guangchun received his Ph.D. and M.S degrees from the University Electronic Science and Technology of China (UESTC) in 2004 and 1999, respectively. He joined the University of California at San Diego and University of Toronto, Canada as a visiting scholar. Currently, he is a professor and Vice President of Graduate School of the UESTC. He has published around 50 articles in international journals and conference proceedings. He is very active in a number of technical societies and serves as committee members of the Calculation of Sichuan Province Institute of High Performance Computer and China Education Information Council. He has received regular funds from many programs of the Chinese Government, including the National Natural Science Foundation of China, the National Hi-Tech R&D Program (the "863" Program), and the Science and Technology Department Foundation of Sichuan Province.

Since the first computer was invented in 1946, a few major milestones have stood along the way of Information Technology's development. Among them, most notable ones are personal computers (PC), the Internet, World Wide Web (WWW), and cloud computing. Cloud computing has drastically changed the landscape of IT industry by providing some major benefits to IT customers: eliminating upfront IT investment, scalability, proportional costs, and so on. In addition, cloud computing also prompts new challenges and brings in new progress. However, the delay-sensitive applications face the problem of large latency, especially when several smart devices are getting involved. Therefore, cloud computing is unable to meet the requirements of low latency, location awareness, and mobility support.

To overcome this problem, Cisco has first introduced a trusted and dependable solution through fog computing to put the services and resources of the cloud closer to users, which facilitates the leveraging of available services and resources in the edge networks. Edge devices such as routers, routing switches, integrated access devices provide an entry point into enterprise or service provider core networks. Fog computing is a scenario where a huge number of heterogeneous ubiquitous and decentralized devices communicate and potentially cooperate with each other and with the network to perform storage and processing tasks without the intervention of third-parties. Similar to cloud computing, fog computing provides data, compute, storage, and application services to end users.

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. Roughly speaking, fog computing mainly involves automation devices while dew computing mainly involves computers; fog computing mainly involves devices such as routers and sensors in the Internet of Things (IoT), while dew computing mainly involves on-premises computers.

The definition and features of cloud computing, fog computing, and dew computing, the relationships among them, and their applications are still under heated discussions and are the focus of this special issue. This special issue aims to provide a unique and timely forum to address all issues and definition related to cloud computing, fog computing, and dew computing. The open call-for-papers of this special issue has attracted excellent submissions in both quality and quantity. After two-round careful reviews, seven excellent papers have been selected for publication in this special issue which is consists of six research papers and one review paper.

The basic idea of fog computing and edge computing is to bring cloud servers closer to end users at the edge of network, reducing the code/data transferring and/or exchanging time between mobile devices and servers. The first paper "A Transparent and User-Centric Approach to Unify Resource Management and Code Scheduling of Local, Edge, and Cloud" by ZHOU et al. addresses this challenging issue. The authors propose a software-defined code scheduling framework which allows the code execution or data storage of end applications to be adaptively done at appropriate machines under the help of a performance and capacity monitoring facility, intelligently improving application performance for end users. The pilot system is described and preliminary performance results are reported.

The proliferation of the Internet of Everything (IoE) has pulled the computing to the edge of the network, such as smart home, autonomous vehicles, robots, and so on. The operating system as the manager of the computing resources, is also facing new challenges. The second paper "An OS for Internet of Everything: Early Experience from A Smart Home Prototype" by CAO et al. presents Sofie, which is a smart operating system for IoE. The authors introduce the design of Sofie, and also show the implementation of Sofie for smart home. Their work shows that Sofie could be helpful for practitioners to better manage their IoE systems.

Cloud operating systems (COSs) working on the physical device and legacy operating system, manage and schedule all hardware and software resources, and provide fundamental resources and operation environment for cloud applications. The third paper "HCOS: A Unified Model and Architecture for Cloud Operating System" by CHEN et al. introduces a unified model and architecture for providing both Infrastructure as a Service (IaaS) and Platform as a Service (PaaS) services.

The fourth paper "Dew Computing and Transition of Internet Computing Paradigms" by WANG et al. focuses on the development of dew computing, including its origins, research status, development status, and its impact on the transition history of Internet computing paradigms. The paper also creates two metrics to measure how important the Internet is and what data redundancy rates are in different stages of the Internet. The discussion and analysis there provide a systematical guideline for beginner researchers in dew computing.

Cloud computing platform, as the mainstream hosting platform for large-scale distributed computing, provides a parallel computing framework for big data job execution. The fifth paper "Online Shuffling with Task Duplication in Cloud" by ZANG and GUO studies minimization of the traffic cost for data pipelining task replications by transmitting non-overlapping data in a cloud computing network. To achieve this goal, the authors present a controller, which chooses the data generated by the first finished task replication and discards data generated later by other task replications belonging to the same task.

With the development of cloud computing, container technology becomes one of the hot issues in recent years. The sixth paper "Technical Analysis of Network Plug-In Flannel for Containers" by YANG et al. studies the technical implementation of the Flannel module, a network plug-in for Docker containers, including its functions, implementation principle, utilization, and performance.

The last paper "Virtualization Technology in Cloud Computing Based Radio Access Networks: A Primer" by ZHANG and PENG surveys the virtualization technology in cloud computing based radio access networks (CC-RAN), focusing on C-RAN, H-CRAN, and F-RAN. The background of network virtualization, virtualization architecture, key enabling technologies, as well as open issues are all discussed. The challenges and open issues mainly focus on virtualization levels for CC-RANs, signaling design for CC-RAN virtualization, performance analysis for CC-RAN virtualization, and network security for virtualized CC-RANs.

Finally, we would like to thank all the authors for contributing their papers to this special issue and the external reviewers for volunteering their time to review the submissions. We would also like to thank the staff at *ZTE Communications* for giving us the opportunity to organize this timely special issue in this esteemed journal and their help during the production of this special issue.

DOI:10.3969/j.issn.1673-5188.2017.04.001
http://kns.cnki.net/kcms/detail/34.1294.TN.20171025.1001.002.html, published online October 25, 2017

A Transparent and User-Centric Approach to Unify Resource Management and Code Scheduling of Local, Edge, and Cloud

ZHOU Yuezhi[1], ZHANG Di[1], and ZHANG Yaoxue[2]
(1. Tsinghua University, Beijing 100084, China;
2. Central South University, Changsha 410083, China)

Abstract

Recently, several novel computing paradigms are proposed, e.g., fog computing and edge computing. In such more decentralized computing paradigms, the location and resource for code execution and data storage of end applications could also be optionally distributed among different places or machines. In this paper, we position that this situation requires a new transparent and user-centric approach to unify the resource management and code scheduling from the perspective of end users. We elaborate our vision and propose a software-defined code scheduling framework. The proposed framework allows the code execution or data storage of end applications to be adaptively done at appropriate machines under the help of a performance and capacity monitoring facility, intelligently improving application performance for end users. A pilot system and preliminary results show the advantage of the framework and thus the advocated vision for end users.

Keywords

cloud computing; fog computing; edge computing; mobile edge computing; resource management and code scheduling

1 Introduction

In the past several years, cloud computing has made a significant achievement by penetrating into our daily work and life through pervasive smart devices and fixed or mobile networks. Large or small companies increasingly adopt commercial clouds (e.g., EC_2 by Amazon, G Suite by Google, and Azure by Microsoft) or private clouds to host their computing or storage services. A cloud typically consists of hundreds or thousands of servers, facilitating flexible and rapid access to a shared pool of dynamically configured and nearly unlimited computational and storage resources. The two major advantages of cloud computing (i.e., the elasticity of resource provisioning and the pay-as-you-go pricing model) enable users to use and pay only for their actually needed resources, inspiring most companies to transfer their traditional computing facilities to cloud platforms.

In the meanwhile, it has witnessed the pervasive usage of smart mobile devices, such as smart phones, pads, and wearables to access thousands of services (e.g., Facebook, Twitter, Wechat) hosted on different cloud platforms. As of 2011, the global shipments of smart mobile devices had exceeded PCs. At the end of 2016, the global mobile users reached up to 7 billion [1]. End users of mobile devices often use the proliferated high-speed wireless access technologies (e.g. long term evolution (LTE), 4G, worldwide interoperability for microwave access (WiMax)) to access services from remote cloud platforms, making cloud computing into a new developing stage. However, even with more powerful computational and storage equipment, the process and energy of mobile devices are still limited due to the fact that mobile applications are now becoming more sophisticated than ever in terms of computing and storage requirements. To alleviate this resource bottleneck of mobile devices, the computation, storage, and networking functions are often offloaded to cloud, giving rise to a new mobile cloud computing (MCC) [2], [3].

However, the relatively poor performance of long-haul wide-area and wireless networks and the unpredictable contention for the shared system resources at consolidated cloud servers make end-user experiences far away from satisfaction. On one hand, limited network bandwidth, time-varying network quality, and long roundtrip time of long-haul wide-area or wireless networks [4], [5] would impair the code/data transmission and responsiveness of users' demands, especially degrading the us-

This work was supported in part by Initiative Scientific Research Program in Tsinghua University under Grant No. 20161080066, and in part by International Science & Technology Cooperation Program of China under Grant No. 2013DFB10070.

er experience of interactive applications. On the other hand, the computation consolidation at the shared resources (e.g. CPU, disks, and I/O) at cloud servers would bring with increased and unpredictable contention, which may lead to significant performance degradation in terms of code execution time and result-return latency. This problem of noisy-neighbors or multitenancy is an outstanding issue especially in public or commercial cloud [6]. Unstable and unpredictable performance due to multitenancy has also been observed when executing computation-intensive scientific tasks on commercial cloud platforms [7] and has even led companies to give up the usage of cloud [8].

Recently, many researchers have proposed that in order to address these challenges faced by traditional cloud computing in various application environments, new computing paradigms have to be brought forward under the principal of "bringing cloud closer to end user". Typical computing paradigms are proposed including fog computing [9], mobile edge computing [10], edge computing [11], and dew computing [12]. This means there is a paradigm shift from the centralization of cloud computing to the decentralization of fog/edge computing similar to the shift from Mainframe computing to PC computing. Considering that the resource management and code scheduling in cloud computing has to be changed due to the paradigm shift from PC computing, we position that the resource management and code scheduling in the post-cloud computing era would also call for new mechanisms and approaches. Obviously, how to manage resources and schedule codes in an efficient and/or optimal manner is very challenging in today's computing environments, considering the co-existence of local, fog, and cloud computing. In this paper, we advocate for a transparent and user-centric approach as a novel mechanism to manage the resources and codes among a local device, its peer devices, nearby fog/edge servers, and cloud servers in a unified manner from an end user viewpoint, and then schedule and distribute the code to execute on the appropriate machines in a transparent and intelligent way under the help of new facilities or tools. We position that this management and scheduling approach will retain the core advantages of using Micro clouds (formed with one or several fog/edge servers) and/or clouds as a support infrastructure but will put back the control and trust decisions to end users in an easy and hales-free way, allowing for novel mobile applications.

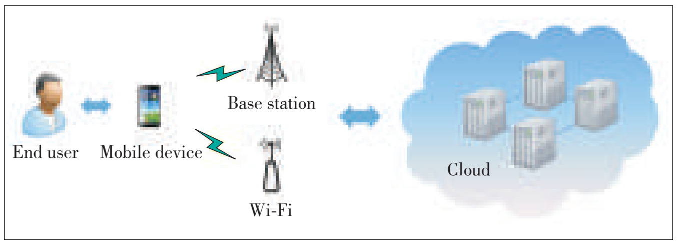
The remainder of this paper is organized as follows. In Section 2, we describe our idea and position of a transparent and user-centric mechanism for resource management and code scheduling in the virtually unified federal computing environment consisting of the local computing, fog/edge computing, and cloud computing. In Section 3, we present a software-defined code scheduling framework for illustration of our vision in details. In Section 4, we discuss a pilot system of the proposed code scheduling framework and evaluate the system through several experiments to demonstrate the effectiveness of the framework. Finally, we conclude the paper in Section 5.

2 Unifying Resource Management and Code Scheduling for Local, Edge, and Cloud

In a typical cloud computing environment, as shown in **Fig. 1**, there is a data center with logically unlimited compute and storage capacity. A lot of end-user devices surround the data center and connect to it with a wireless and/or long-haul wide-area network and access the services from the cloud. An end user manages the local resources of their devices (e.g., smartphones, iPads) and run their applications on the local machine. If needed, the end user requires to use the compute or storage resources from the data center. The cloud provider or the system administrator of the data center manages and schedules the resources of the cloud platform based on their benefits in a centralized manner and chooses the policy or strategy about where and how to execute the offloaded code from end users.

If the local compute resource is not enough for the end-user device, an end user can upload/offload these computation-intensive codes and related data to the cloud to execute there. Usually, two steps may occur during the period of the code execution. In the first placement (assignment) step, the uploaded codes is put on a specific server chosen in the data center by the code scheduler or dispatcher of the cloud platform based on current system status of resource usage. However, if the system status is changed due to some reasons (e.g., underutilization of server resources, conflicts of other resource consumers), the second reassignment (also referred to as migration in virtual machine (VM) based code offloading) step will occur and the code scheduler will transfer and schedule the code (accompanying with its data) to run on another more appropriate or optimal server in the data center. Note that this reassignment step to achieve an optimal performance or resource utilization may occur multiple time during the period of the code execution based on the tradeoff between the quality of experience (QoE) of end users and the resource operating costs/polices and relative benefits of the cloud provider.

Due to the natural ownership of the local end-user devices and the remote cloud servers, the resource management and code execution scheduling policies and strategy are made independently by end users and the cloud owner or provider based on their respective benefits and the service level agreement (SLA). Obviously, in the above centralized cloud computing



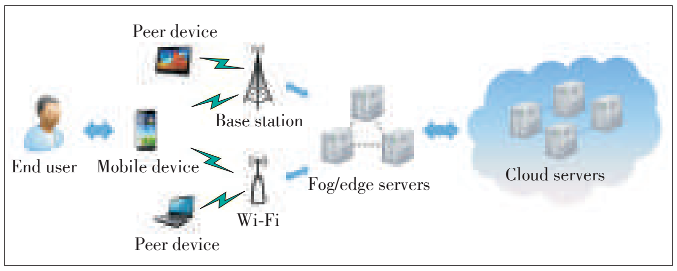
▲**Figure 1. Typical scenario of a cloud computing system.**

system, centralized resource management and code scheduling for offloaded codes can make usage of the resources of cloud platform by fully exploiting consolidation to improve the resource utilization and reduce the operating cost in the data center. It is also reasonable for end users to just require and demand the execution service based on the SLA. Therefore, the current centralized resource management and code scheduling without the involvement of end users is natural and suitable for cloud computing-based systems.

On the contrary, with the recently emerged fog/edge computing, there might be servers at nearby Nano data centers closer to end users in addition to servers located in remote data centers. Considering that fog/edge computing is just an extension of cloud computing at the edge of network, we believe that the traditional centralized cloud computing and the newly decentralized fog/edge computing will coexist for a long time. In such a situation, there would be several types of machines or places that end users can put their application codes to execute, including the local device, peer devices, nearby fog/edge servers, and remote cloud servers.

As shown in **Fig. 2**, we consider a general picture of mixed or federal computing environments in which end users can offload their computing tasks/codes to different computing devices, including local mobile devices, peer mobile devices, nearby fog/edge servers, and remote cloud servers in data centers. All of these computing devices or servers are interconnected through wireline or wireless links. Typically, the access link from the mobile devices to the fog/edge server is wireless. Generally, fog/edge servers are attached to the points of wireless access in a form of Nano data center or Micro cloud. Nearby Micro clouds of fog/edge are generally characterized by limited computation or storage resources, but are directly accessible by end users to avoid the additional communication delay of the Internet. Considering the cost of establishing a Micro cloud of fog/edge could be very little, even with only a single server, we believe that there could be multiple nearby Micro clouds around an end user and these clouds may be or not be owned and managed by the same provider.

Obviously, a mixed computing environment consisting of local devices, peer devices, edge servers, and cloud servers is much more complex compared to a single cloud environment of local devices and cloud servers. It is natural that the resource management and code scheduling mechanism in a mixed scenario would be much more complicated to achieve a global optimization. Currently, a cloud provider only commits to availability of their services, through SLAs with their clients. Therefore, there are numerous inherent factors that introduce uncertainty regarding the actual performance of an offloaded task by end users in a cloud computing infrastructure [13]. However, in a mixed computing environment, if the performance of an offloaded task degrades due to the uncertainty of a cloud server, an ideal code scheduler or dispatcher can reassign its code to execute on another fog/edge server in the whole federal computing system to exploit better resources than that of any server in the original cloud to furtherly reduce the execution time and mitigate the effects of uncertainty than before. This is the basic reason that we advocate for a new and unified resource management and code scheduling mechanism for efficiently and optimally utilizing the whole system resources across different computing devices and different Micro clouds or cloud servers in the post-cloud era. This will bring end users with better performance and much more satisfactory experience.



▲**Figure 2. Typical scenario of a mixed computing environment.**

In this section, we elaborate the resource management and code scheduling mechanism that enables end users to leverage the resources that across all of the computing devices and servers that may be owned and managed by different individuals or organizations. We positon that end users can select and choose where and how to execute their offloaded applications (e.g., code, data, networking) at their will and benefits. In contrast to the centralized approach where the cloud provider selects the cloud-wide optimal scheduling strategy, we advocate for transparent and user-centric approaches where the decisions are made independently by each end user or task. First, we delineate the performance benefits that arise for mobile end users in emerging computing paradigms and identify the opportunities to logically unify the resource management and code scheduling across different computing devices, Micro clouds and cloud servers to improve the performance in a user-centric viewpoint. Second, we explain the difficulties and challenges for end users to manage and schedule their tasks/codes across different computing devices, Micro clouds and/or cloud servers, and identify the requirement that an intelligent facility or tool is called for to help end users and even Micro cloud or cloud providers to achieve an optimal scheduling for them under several constraints to achieve a more efficient resource utilization among the whole mixed and federal system.

2.1 Unifying Resource Management and Code Scheduling from User-Centric Viewpoint

As stated above, the resource management and code scheduling mechanism in a cloud is a centralized one where the cloud provider selects the scheduling strategy for the cloud based on the benefits of their own. Obviously, this mechanism is a cloud-centric or server-centric approach. Because the cloud is a dynamically changing environment, its performance depends on numerous uncontrollable and unpredictable parameters. The performance of an offloaded task/code cannot be guaranteed and it may upgrade or degrade with the available resources on the hosted server. However, in a mixed computing environment of local, edge, and cloud, the current cloud-centric approach cannot utilize the resources across different computing devices, Micro clouds or cloud servers as a whole and thus cannot globally achieve an optimal scheduling strategy from the perspective of an end user. To elaborate the benefits of a new code scheduling mechanism, we give two hypothetical application scenarios based on code offloading of mobile applications.

In Scenario 1, because of his smart phone's limited computing power, Bob offloads a computation-intensive task into a server (S1) in a remote cloud owned and managed by the provider P1 to efficiently execute the code according to the SLA. Afterwards, he moves to another place where there is a nearby fog server (F1) that is free and its available computing power is much more than the host server S1 that is now competitive for several tasks of other end users.

In Scenario 2, Alice wants to offload her computation-intensive task into a nearby edge server due to her limited smart phone's computing power. There are two edge severs (E1 and E2) owned and managed by two providers (P2 and P3) respectively. Without further information, she chooses the cheaper E1 to execute the task. Afterwards, another woman also chooses the E1 to execute her code, reducing the computing power shared by Alice. At the same time, E2 is available with a much higher computing capacity.

With the current cloud-centric or server-centric code scheduling mechanism, there is not a facility to help Bob and Alice to schedule their codes across Micro clouds or clouds, thus what Bob and Alice can do is only waiting for his or her task to finish without any control of the execution of their offloaded tasks.

If we imagine that there is an ideal global resource management and code scheduling facility or tool that can obtain the information online about dynamics of the computing device, Micro cloud and cloud server, with the help of such a facility, Bob could move his task to F1 and Alice could reassign her task at E2 to leverage the more powerful computation resources to execute their tasks. Here we assume that the overhead of code and data transferring can be omitted compared with the achieved performance and the network link capacity and characteristics should be kept at a same level.

Based on the above observations and considerations, we advocate that novel resource management and code scheduling mechanisms are required to improve the performance of offloaded functions or tasks for end users and/or to achieve a more efficient global resource utilization. Here, we move to distributed approaches where the scheduling decisions are made independently by an end user or each offloaded function/task. To adhere to a realistic scenario, we assume that information about the dynamics of computing devices, Micro clouds and cloud servers is presented online, not available as a priori, just before a scheduling decision is made. Therefore, from the perspective of an end user, they can obtain these information to choose or select optimal or near-optimal machines to execute their functions/tasks. We also assume that the code/data uploading (transferring) or task migration process are standardized and can be carried out across different computing devices, Micro clouds, and clouds to form a virtually federal computing system. Considering the reality of industry standardization and industry alliance, we think this assumption is also realistic.

Therefore, we take a user-centric approach to schedule the tasks/codes to run at the selected optimal places or machines based on the benefits of end users from time to time, not just of cloud providers. In this way, the resource management and code scheduling of the whole system consisting of local, edge, and cloud is logically unified from the viewpoint of an end user. Note that the computing devices, Micro clouds or cloud servers are still owned and managed by their owners and they just make a logically federal computing system by interactions and agreements.

From a user-centric view, the virtually unified Internet computing environment consisting of data centers and clouds is at the core, Nano data centers (consisting of standard servers or routers, Wi-Fi points, and base stations with available computing capacities) and Micro clouds at the edge, and peer computing devices such as desktop PCs, tablets, and smart phones at the leaf. Based on this view, the resource management and code scheduling mechanism helps end users to leverage the resources of such a virtual system as a whole and achieve the most benefits. In summary, a user-centric approach to resource management and code scheduling encompasses the following elements:

1) Human-centered design: Human-centered design considers humans as an important factor in designing and developing a mechanism. First, humans should naturally interact with the system and impact the system behavior. Second, humans should be put in the control loop, so that users can take or retake control of their information. This leads to the design of novel crowdsourced and informed scheme where users control the information flows. Finally, the human-centered design also calls for novel and innovative forms of human-centered applications.
2) User-controlled decisions: The decision-making about the resource management and code scheduling is under the full control of end users. First, end users can choose and obtain all needed information about their nearby peer devices, fog/edge servers, and/or cloud servers. Second, end users are able to freely coordinate with computing devices or servers surrounding them to assign or delegate computation synchronization or storage to other computing devices or servers selectively. Finally, end users can get the status of their offloaded tasks and decide where and how to reassign their tasks to another computing device or server based on their

own intentions.

3) Local or edge proximity: It is more efficient to communicate and distribute data among nearby computing devices or servers than to use resources far away from end users. Here, of course, nearby can be understood both in a physical sense and a logical sense.

4) Local or edge trust: Personal and social sensitive data is clearly located in the local or the edge under the control of end users. Therefore, the control of trust relation and the management of sensitive information flows in a secure and private way must also belong to the end user.

2.2 Unifying Resource Management and Code Scheduling in a Transparent Manner

In Section 2.1, we advocate for a novel resource management and code scheduling mechanism that virtually unify and schedule the resources across the local, peer, edge, and cloud and leverage these resources to execute functions or tasks for an end user. However, even with the information about Micro cloud dynamic or cloud dynamic is acquired, it is still very hard for a common end user to choose and select where to upload and execute their codes. Therefore, an intelligent facility or tool should be developed to automate the assignment or reassignment process of offloaded tasks on behalf of end users, in order to make an easy and transparent manner to interact with such a virtual and complex system. Several research efforts have been done to enable an easy and transparent manner to use emerging complex computing systems [14], [15]. Here, we follow the concept of transparent computing as a basis for developing the unified resource management and code scheduling mechanism. To sum up, a transparent approach to resource management and code scheduling encompasses the following elements:

1) Easy-to-use interface: The interface for end users should be easy to use, at least keeping the same experience of offloading tasks to a cloud. Novel and easy-to-use interface are called for to make the code scheduling across different types of computing devices and servers much easier.

2) Hassle-free management: End users should just focus on their tasks, not the techniques about the installation, maintenance, and management of such a scheduling mechanism. This means that the techniques need to be transparent to end users and make the complex system of systems as a single system.

3) Intelligent decision-making: As it is very hard for humans to handle the information obtained from time to time and make a decision where and how to schedule the codes to execute, the novel mechanism or approach could semi-autonomously or autonomously help end users to make decisions of placement or reassignment according to their requirements. Of course, end users can intervene the process by some means.

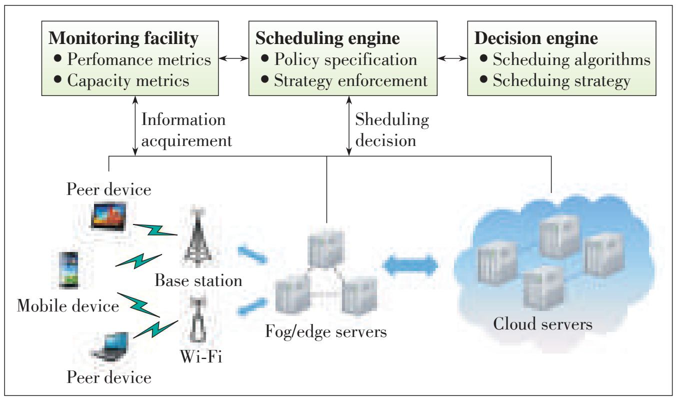
4) Adaptive reaction: Due to the continuous changing characteristics of cloud servers, the new mechanism should also adaptively react to the changed environment. First, it should sense the changes of the hosted servers and the candidate servers that could be reassigned. Second, it should make a decision weather a reassignment is needed. Finally, it should select and reassign the code to another appropriate server to finish the remaining task.

3 Software-Defined Code Scheduling Framework with Capacity and Performance Monitoring

In this section, we illustrate our vision of a novel transparent and user-centric approach to resource management and code scheduling by presenting a software-defined code scheduling framework with capacity and performance monitoring facilities as a case study. As shown in **Fig. 3**, the framework has three parts: the monitoring facilities (MF), scheduling engine (SE), and decision engine (DE). The MF periodically acquires the information about task performance and capacity of any machines in the concerned scheduling domain. If an end user wants to initiate a scheduling process, the SE gets the needed inforamtion from the MF and sends the predefined scheduling policy and the related information to the intellegent DE. The DE makes a scheduling strategy by using various intelligent scheduling algorithms. An end user can initiate a scheduling process periodically or by giving an instruction through defining the scheduling policy for her/his tasks, e.g., the task performance degrading to an unsatisfying level.

To meet diverse scheduling requirements of various kinds of end users, end applications, and usage scenarios, the scheduling polices for different end users or tasks could be defined and specified in advance by a graphical tool or a specific police-defining language.

The framework provides various selections and places for computation and storage of end applications. This characteristic can be used to improve the performance of offloaded task and thus the user experience of end users by assigning computation and storage at appropriate machines. For example, the



▲Figure 3. Software-defined scheduling framework with monitoring facility.

computation can be carried out at four kinds of machines, i.e., *L*, *P*, *E*, *C*, which denote the local device, peer device, fog/edge server, and cloud server, respectively.

Given a computing task, the computation time of the task would be the sum of time consumed by different stages. Typically, there are five stages of the life of a task as follows.

1) Preparation: Once a new task is generated by an end user, the scheduler or dispatcher will choose one or several destination peer devices, Micro clouds, or clouds for her/him. Then the needed code and any input data required for the initialization of the task are transferred to the destinations. Finally, the task will be executed at the selected peer devices, Micro clouds, or clouds.
2) Assignment: Once the required code and data have been uploaded to the destination, a local scheduler or dispatcher is responsible for the assignment of the task to a specific process at the peer devices, or a specific fog/edge or cloud server.
3) Execution: This is the actual processing of the offloaded task/code. A new execution entity, such as a concrete process or a virtual machine is created for the specific task at the selected machine, and it starts to execute immediately.
4) Reassignment: The instance of the executed task/code may be transferred from its current machine to a new one to continue execution there for various reasons. As mentioned before, this reassignment stage may occur multiple times during the task execution. For the further execution of the task, the accompanying related data required for the initialization of the new execution instance has to be transferred to the new destination machine. Note that the reassignment, though optional, may occur several times during the lifetime of a task and may cross different computing devices, Micro clouds or clouds.
5) Outcome: At this stage the mobile end user retrieves the final results. We assume that once the whole processing is completed, the final data are immediately downloaded by the user through the current access technology or forwarded to another place to store.

The total time of finishing the offloaded task is the sum of the time spent at these different stages. Obviously, it depends on several factors. First, the selected machine at the assignment or reassignment stage mainly affects the execution performance because different computing devices or servers have different compute and storage capacity. Second, the constantly changing capacity of the selected machine due to the execution of other offloaded tasks also has an impact on the considered tasks. Third, the scheduled reassignment stage itself may significantly affect the execution due to the needed transferring of codes or data. Thus, the scheduling policy of reassignment, the times of the reassignment, and the selected reassigned machine would have an impact. Finally, the mobility of the end user may make the original scheduling decision inefficient or invalid. All of these factors, even if possible, are very hard to accurately predict and make an optimal scheduling decision. Therefore, it is very challenging to ensure that a scheduling scheme for an offloaded computation task is an optimal scheduling. Considering the situations in the real world, a realistic solution must be given on time to respond to the scheduling demand. We call for further researches to deal with these new challenges in the post-cloud era.

3.1 Monitoring Facilities of Capacity and Performance

If an end user or task initiates a scheduling periodically or due to the observation of performance degradation, the first thing to do for the framework is to learn about the current status of the entire system.

To collect performance metrics of the computation task and the capacity of the peer devices, nearby fog/edge servers, or cloud severs, the MF employs a proxy that resides inside the local and peer devices, nearby fog/edge servers or cloud servers. These proxies periodically collect the performance metrics of computation tasks and the capacity metrics of related candidate destination machines (e.g., CPU and memory capacity and utilization) and then send all the needed information to the SE to form a scheduling decision.

The selection of metrics is critical for the usability of a monitoring facility. The low-level metrics of the physical machines, such as CPU and memory utilization, are easy to collect, but they do not meet the requirements of some types of applications. The high-level metrics of software applications, such as click response, are difficult to collect and specific to certain types of applications. To explore, we develop a tool to choose the appropriate metrics by using a machine learning method based on historical usage data or conducting real-world experiments under different usage scenarios. These metrics can be specified automatically or manually by system administrators.

3.2 Software-Defined and Adaptive Code Scheduling

With the assistance of the above monitoring facilities, we develop a software-defined scheduling engine that can schedule the code in a predefined way or adapt to the changing computing environment based on the monitoring of the task performance in a timely manner. For example, if the performance metric of the monitored task is below a threshold, then the scheduling engine could be triggered, and the scheduling policy predefined by end users would be read and analyzed. Then, all the needed information is sent to the decision engine to form a scheduling strategy, which includes where and how to execute the computation or storage. The decision engine runs a scheduling algorithm and then obtains a scheduling strategy based on the current system status according to the predefined scheduling policy or adaptively learns a scheduling policy based on artificial intelligent rules from the changes occurred in the whole system status before. After getting a scheduling strategy from the decision engine, the scheduling engine takes actions to enforce the scheduling strategy by interacting with the selected peer devices, Micro clouds, or clouds. The whole scheduling process is illustrated by **Algorithm 1**.

Algorithm 1: Software-defined code scheduling process

```
Input:
 The set of performance metrics: $M_1$, $M_2$, ..., $M_n$
 The specification file of scheduling policy: $F$;
Output:
 Scheduling strategy $S$;

Step 1: Retrieve and analyze monitored metrics
Initialize performance metrics: $M_1$, $M_2$, ..., $M_n \Leftarrow$ NULL;
for each i $\in [1, n]$ do
  if $M_i$ is defined then
      Obtain $M_i$ from MF;
  end if
end for

Step 2: Retrieve and analyze scheduling policy

initialize scheduling condition: $Con \Leftarrow$ NULL;
retrieve $F$;
for each i $\in [1, n]$ do
  if $M_i$ is defined in $F$ then
      $Con$ = Con $\cup$ $M_i$;
  end if
end for

Step 3: Judge and select the predefined scheduling algorithm
if $Con \neq$ NULL then
    retrieve scheduling algorithm $A$ from $F$;
    return $A$;
else
    return NULL;
end if

Step 4: Form a scheduling strategy
if $Con \neq$ NULL and $A \neq$ NULL then
    run scheduling algorithm $A$;
    return $S$;
else
    return NULL;
end if
```

Algorithm 2 gives a general example of scheduling algorithm for the proposed framework. Considering that obtaining the optimal placement is time consuming and unnecessary in the real world, several heuristic algorithms are designed for the decision engine to run under different usage scenarios. The end user can give a rule to run any scheduling algorithm by specifying the scheduling policy in the scheduling framework. Therefore, the scheduling is software-defined and controlled

Algorithm 2: A general code scheduling algorithm

```
Input:
 The performance metric: $M_n$
 The threshold of the performance metric: $Tre$;
Output:
 Place to execute the code: D;

Step 1: Estimate the performance at different machines
initialize the set of place candidates: $Pls \Leftarrow$ NULL;
for each $i \in L, P, N, C$ do
  for each $j \in N, C$ do
    Estimate the code preparation performance $T_R^j$;
    Estimate the code transportation performance $T_T^{j,i}$;
    Estimate the code execution performance $T_E^i$;
    Calculate the whole performance $T^{i,j} = T_R^j + T_T^{j,i} + T_E^i$
    if $T^{i,j} \geqslant Tre$ then $Pls = Pls \cup \{i, j\}$
    end if
  end for
end for

Step 2: Choose the best place to execute the code
if $Pls \neq$ NULL then
  Choose the best performance candidate from $Pls$;
  return $\{i, j\}$;
else
  return NULL;
end if
```

by the current system status and the enforced scheduling policy. **Algorithm 3** gives a heuristic example of scheduling algorithm. Here, the algorithm just uses the network performance to predict the actual execution performance of the scheduled

Algorithm 3: A simple heuristic code scheduling algorithm

```
Input:
  The network performance metric: $M_{net}$
  The threshold of the metric: $Tre$;
Output:
  Place to run the code: $D$;

Step 1: Estimate the performance at different machines
initialize the set of place candidates: $Places \Leftarrow$ NULL;
for each $i \in L, P, N, C$ do
  Estimate the value of M net to $Net^i$;
  if $Net^i \geqslant Tre$ then $Places = Places \cup i$
  end if
end for

Step 2: Choose the best place to run the code
```

```
if $Pls \neq$ NULL then
    Choose the best performance candidate from Places;
    return $i$;
else
    return NULL;
end if
```

code, significantly simplifying the implementation complexity. Note that the network performance in the exemplary scheduling policy and the scheduling algorithm (Algorithm 3) is just for illustration and not specified. If the network performance means network bandwidth, the condition judgement of $Net^i \geqslant Tre$ is established. However, if it means network delay, the condition may be set as $Net^i \leqslant Tre$.

4 Pilot System and Preliminary Results

4.1 Pilot System

We developed a pilot system using commercial off-the-shelf desktop machines and servers to demonstrate the feasibility and effectiveness of the proposed software-defined code scheduling framework based on the transparent computing [15].

The pilot system uses VM technology to encapsulate the codes of OSes and their applications. The computation or storage can be scheduled and executed on three types of machines or servers, namely, local machines, nearby fog/edge servers, and remote cloud servers.

To simplify the implementation, a heuristic threshold-based scheduling algorithm is designed to run when the performance is below a predefined threshold. More details about the pilot system can be found in [16].

4.2 Preliminary Results

Two sets of experiments are conducted to show the effectiveness of the proposed framework. In all experiments, local machines use Intel Core i7-4790 (8 cores, 3.6 GHz) chips, with 12 GB DDR3 RAM, a 1TB Seagate 7200 rpm hard disk, and a 1 Gbps Realtek RTL8139C (plus Fast Ethernet) NIC. The nearby fog/edge servers use Intel Xeon e5-2620 (24 cores, 2.0 GHz) chips, with 24 GB DDR3 1333 RAM, one 1 TB Western Digital 7200 rpm RAID5 hard disk, and a 1 Gbps NetXtreme BCM 5720 Gigabit Ethernet network card. Local machines and nearby servers are connected by a TP-Link TLSG-1048 Ethernet switch with 48 1 Gbps ports. The OS and software code are encapsulated as VMs and scheduled to run at the local machines, nearby servers, and Amazon cloud servers, which are configured with one virtual CPU, with 2 GB memory and a 50 GB hard disk. The nearby servers use XenServer 6.0.2 OS. The local machines use Ubuntu 14.04, and a VM based on VirtualBox 4.3.36 runs on every machine. The VMs run Windows 7 SP1 with 1024 × 768 (32-bit color depth) resolution. The local machines access the VM running on the nearby server through SSH virtual network computing (VNC) 1.0.29 and access the VM running on the Amazon WorkSpace cloud server through Amazon WorkSpace 2.0.8205 client.

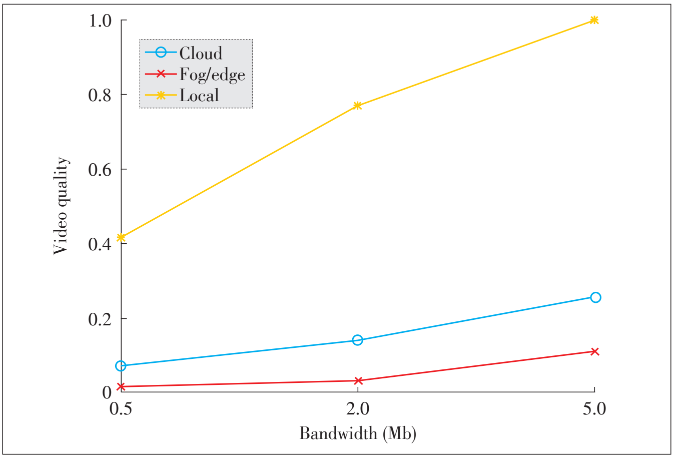
First, we evaluate the performance under different scenarios with standard 2D graphics test suite of PassMark Performance Test 8.0. This test suite includes operations of drawing lines, bitmaps, fonts, text, and GUI elements. To be more specific, eight tests are: simple vector, complex vector, fonts and text, Windows interface, image filters, image rendering, direct 2D, and overall 2D performance. We also compare the performance in three scenarios: scheduling code to run at the remote Amazon cloud server, the fog/edge server (accessing with VNC), and the local machine.

As shown in **Table 1**, the overall 2D performance of executing code on local machines outperforms that on the nearby fog/edge server and the fog/edge outperforms the cloud. This means that the performance is worse when the software codes are executed farther away from the end user. However, in the test of Windows interface, the fog/edge gets a slightly better performance than the other two, and in the test of image rendering, the cloud achieves the best performance. These two tests show that the performance will be better if the codes are scheduled to run at appropriate machines.

Second, we evaluate the video playback performance by playing a 21-second video clip, which has 320 × 240 pixels and 24 frames per second, in the 1024 × 768 resolution with Windows Media Player 12.0. The performance is measured with slow-motion benchmarking [17], which uses a packet monitor to capture network traffic for quantifing performance in a non-invasive manner. The video playback quality defined in the slow-motion benchmarking is taken as the performance metric. The results are shown in **Fig. 4**. We evaluate the performance under different scheduling strategies. In each scheduling strategy, the network bandwidth is changed to see how it affects the video quality. The video quality of the local machine with 2 Mb bandwidth (nearly 1) is better than that of the fog/edge server (nearly 0.11 with VNC) and the remote Cloud server (nearly 0.25) when the codes are scheduled to run at the local machine. This result indicates that adaptive scheduling can

▼Table 1. Performance of 2D graphics (Scores)

Test	Cloud	Fog/Edge	Local
Simple vector	29	28.6	38.8
Complex vector	109.1	171.6	191.5
Fronts and text	136.1	151.7	159.9
Windows interface	115.5	116.6	109
Image filters	608	672	725
Image rendering	575	513	507
Direct 2D	6.6	8.5	13.8
Overall 2D Performance	481.5	554	659



▲Figure 4. Video quality performance.

sharply enhance video playback quality, and such enhancement substantially improves the end-user experience. A comparison of the video quality with different network bandwidths shows that the video quality is highly sensitive to the bandwidth. This also explains that our adaptive scheduling framework can achieve better performance by scheduling the software codes to run at machines with higher network bandwidth to end users.

5 Conclusions

The recently emerging computing paradigms, such as fog/edge computing, have brought great changes to the computing environments for mobile end users and also numerous opportunities for creative applications. However, the current cloud-centric or server-centric resource management and code scheduling mechanisms are not suitable in the complex computing environment that consists of different Micro clouds or clouds. Thus, we advocate for new and novel resource management and code scheduling mechanism to deal with such a challenge and elaborate the details of this vision. To illustrate this vision, we proposed a software-defined scheduling framework to assign or reassign computation and/or storage to appropriate machines, including peer computing devices, nearby fog/edge servers and remote cloud servers. To demonstrate the effectiveness in real world, we also developed a pilot system. The pilot system shows that our vision and approach have good potential to further improve the performance of end applications and user experience in today's emerging computing environments by adaptively scheduling the codes to execute on appropriate machines. Further research and exploration are also called for realizing such a transparent and user-centric approach to globally achieve efficiency of resource utilization and optimal performance of code execution.

References

[1] B. Sanou. (2017, July 22). ICT Facts and Figures 2016 [Online]. Available: http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2016.pdf
[2] H. Flores, P. Hui, S. Tarkoma, et al., "Mobile code offloading: from concept to practice and beyond," *IEEE Communication Magazine*, vol. 53, no. 3, pp. 80–88, Mar. 2015. doi: 10.1109/MCOM.2015.7060486.
[3] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: a survey," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 84–106, Jan. 2013. doi: 10.1016/j.future.2012.05.023.
[4] D. Chu, A. Kansal, J. Liu, and F. Zhao, "Mobile apps: it's time to move up to condos," in *Proc. 13th Usenix Conference on Hot Topics in Operating Systems*, California, USA, May 2011, pp. 16–16.
[5] N. Tolia, D. Andersen, and M. Satyanarayanan, "Quantifying interactive experience on thin clients," *Computer*, vol. 39, no. 3, pp. 46–52, Mar. 2006. doi: 10.1109/MC.2006.101.
[6] A. Williamson (2010, Jan. 12). Has Amazon EC_2 Become over Subscribed [Online]. Available: http://alan.blog-city.com/has_amazon_ec2_become_over_subscribed.htm
[7] A. Iosup, S. Ostermann, M. N. Yigitbasi, et al., "Performance analysis of cloud computing services for many-tasks scientific computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 6, pp. 931–945, Jun. 2011. doi: 10.1109/TPDS.2011.66.
[8] Mixpanel Engineering. (2011, Oct. 27). Mixpanel: Why We Moved Off the Cloud [Online]. Available: https://code.mixpanel.com/2011/10/27/why-we-moved-off-the-cloud
[9] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proc. MCC' 2012*, New York, USA, Aug. 2012, pp. 13–16. doi:10.1145/2342509.2342513.
[10] H. Li, G. Shou, Y. Hu, and Z. Guo, "Mobile edge computing: progress and challenges, " in *Proc. 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, Oxford, UK, Mar. 2016, pp. 83–84. doi: 10.1109/MobileCloud.2016.16.
[11] W. S. Shi, J. Cao, Q. Zhang, et al., "Edge computing: vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Jun. 2016. doi: 10.1109/JIOT.2016.2579198.
[12] Y. W. Wang. (2015, Nov. 10). The Initial Definition of Dew Computing [Online]. Available: http://www.dewcomputing.org/index.php/2015/11/10/the-initial-definition-of-dew-computing
[13] L. Gkatzikis and I. Koutsopoulos, "Migrate or not? exploiting dynamic task migration in mobile cloud computing systems," *IEEE Wireless Communications*, vol. 20, no. 3, pp. 24–32, Jun. 2013. doi: 10.1109/MWC.2013.6549280.
[14] R. Murch, *Autonomic Computing*. USA: IBM Press, 2004.
[15] Y. Zhang, K. Guo, J. Ren, et al., "Transparent computing: a promising network computing paradigm," *IEEE/AIP Computing in Science & Engineering*, vol. 19, no. 1, pp. 7–20, Jan. – Feb. 2017. doi: 10.1109/MCSE.2017.17.
[16] Y. Zhou, W. Tang, D. Zhang, and Y. Zhang, "Software-defined streaming-based code scheduling for transparent computing," in *Proc. 2016 International Conference on CBD*, Chengdu, China, Aug. 2016, pp. 296–303. doi: 10.1109/CBD.2016.058.
[17] J. Nieh, S. J. Yang, and N. Novik, "Measuring thin-client performance using slow-motion benchmarking," *ACM Transactions on Computer Systems*, vol. 21, no. 1, pp. 87–115, Feb. 2003. doi: 10.1145/592637.592640.

Manuscript received: 2017-05-10

Biographies

ZHOU Yuezhi (zhouyz@mail.tsinghua.edu.cn) is an associate professor in the Department of Computer Science and Technology, Tsinghua University, China. His research interests include distributed system, mobile network, and transparent computing system.

ZHANG Di (dizhang@tsinghua.edu.cn) is a postdoctoral researcher in the Department of Computer Science and Technology, Tsinghua University, China. His research interests include distributed system, mobile network, and transparent computing system.

ZHANG Yaoxue (zyx@csu.edu.cn) is a professor in the School of Information Science and Engineering, Central South University, China. His research interests include transparent computing, pervasive computing, and big data.

DOI: 10.3969/j.issn.1673-5188.2017.04.002
http://kns.cnki.net/kcms/detail/34.1294.TN.20171016.1814.002.html, published online October 16, 2017

An OS for Internet of Everything: Early Experience from A Smart Home Prototype

CAO Jie, XU Lanyu, Raef Abdallah, and SHI Weisong
(Department of Computer Science, Wayne State Unversity, Detroit, MI 48201, USA)

Abstract

The proliferation of the Internet of Everything (IoE) has pulled computing to the edge of the network, such as smart homes, autonomous vehicles, robots, and so on. The operating system as the manager of the computing resources, is also facing new challenges. For IoE systems and applications, an innovative operating system is missing to support services, collect data, and manage the things. However, IoE applications are all around us and increasingly becoming a necessity rather than a luxury. Therefore, it is important that the process of configuring and adding devices to the IoE is not a complex one. The ease of installation, operation, and maintenance of devices on the network unarguably plays an important role in the wide spread use of IoE devices in smart homes and everywhere else. In this paper, we propose Sofie, which is a smart operating system for the IoE. We also give the design of Sofie. Sofie can be implemented via different IoT systems, such as Home Assistant, openHAB, and so on. In order to implement Sofie to get some early experience, we leverage Home Assistant to build a prototype for the smart home. Our work shows that Sofie could be helpful for practitioners to better manage their IoE systems.

Keywords

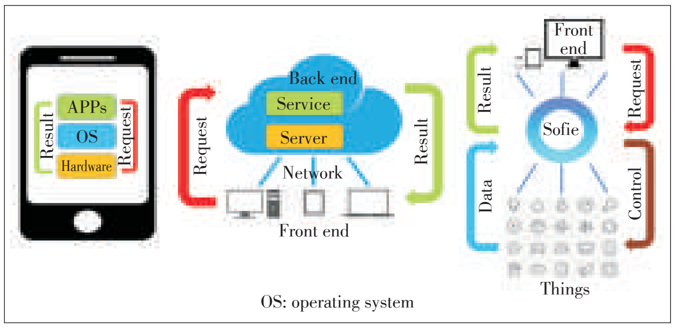
edge computing; IoE; smart home; operating system

1 Introduction

With the burgeoning of the Internet of Everything (IoE), computing in our lives is shifting from PC, mobile device, and cloud to the things at the edge of the network [1]. More and more data is generated as well as consumed at the edge of the network. According to the report from Cisco Global Cloud Index, the data produced by people, machines, and things will reach 500 zettabytes by year 2019 [2]. Moreover, 45% of the data contributed by the things will be stored, processed, analyzed, and acted upon close to, or at the edge of the network [3]. While more devices and applications are coming out for IoE, the operating system (OS) for IoE is still unavailable. Conventional operating systems for PC and smart phones care more about resource management. The application practitioners directly use hardware resources since the hardware of a computing platform is well-designed and forms a fixed environment. For cloud computing, a service provider will manage all the hardware so that application practitioners only need to focus on the service. In this case, cloud computing OS is designed as a service-oriented architecture, and various members of the "as a Service" (aaS) family such as Software as a Service (SaaS), Infrastructure as a Service (IaaS), and Platform as a Service (PaaS).

The IoE operating system should have similar functions as traditional operating systems; however, there are several differences. In the IoE, similar as cloud computing, application practitioners should not care about the hardware. However, unlike cloud computing system that has service-oriented architecture, an IoE operating system should be data-oriented, considering that most of the things are just passively collected and the reported data are in a predefined manner. Moreover, IoE operating system should also be responsible for hardware management like PC or smart phone operating systems. This is because the IoE is a highly dynamic system where devices are added or removed frequently.

Therefore, traditional OS is not suitable for the IoE. In this paper we introduce a smart operating system for the IoE—Sofie. In **Fig. 1**, we compare the mobile operating system, cloud, and Sofie. In the case of mobile device and PC, the operating system focuses more on resource management and provides interface for applications to access hardware resources since the software and hardware are all fixed in a determinate machine. When the applications want to access any hardware resource, they can directly send a request to the operating system. On the other hand, in the case of cloud computing, the users on the front end might have limited information about hardware re-



▲Figure 1. Comparison of mobile operating system, cloud, and Sofie.

sources on the other end of the network. Therefore, the cloud service providers should access the request and send a computing result back to the users in a distributed manner. For IoE, Sofie faces new challenges [4]–[8]. Unlike the mobile, PC, and cloud, where the hardware configuration is fixed to a certain extent, the things in IoE could be highly dynamic and unreliable. Moreover, the things, which compose the back end of the paradigm, usually have very limited computing resources (just passive report data and receive commands). To better serve the users on the front end, Sofie should be able to abstract the data from the things and satisfy the front end by taking the computing back to Sofie, rather than forward the request to the things on the back end.

Sofie is both data-oriented and things-oriented. The detailed design of Sofie will be presented in the following several sections. In this paper, we would like to take smart homes as an example to show how to implement Sofie in a domestic environment and how Sofie works to help practitioners better manage their IoE environments.

We organize the remainder of this article as follows: In Section 2, we present the design of Sofie with things management and data management as two major functions. We further show how Sofie looks like in a real IoE environment. We choose a home as an environment for Sofie and present the architecture of Sofie in Section 3. In Section 4, we review both commercial and open source products for smart homes, and evaluate them from different aspects. We introduce the implementation of Sofie on top of Home Assistant and also show how Sofie works for smart homes in Section 5. In Section 6 we present the lessons learned through our work experience with smart homes. Finally, the paper concludes in Section 7.

2 Design of Sofie

In the IoE, although there are various kinds of sensors and devices that can produce data, they work in a passive manner of data reporting, without adopting any action to process the data [9]– [11]. However, for practitioners, the generated data could be leveraged for multipurpose use in the society. Such utilization fields include video analysis, smart homes, smart cities, connected health, and more. All these fields treat data as an indispensable ingredient. For example, data produced in a smart home is consumed to improve the user experience for the occupants; data captured by traffic cameras is retrieved to track suspicious vehicles; data provided by the police department or city hall is utilized to benefit the public; data collected in connected health is used to facilitate communication among hospitals, pharmacies, insurance companies and so on.

Given the significance of data in IoE applications and systems, as well as the underlying hardware, a data-driven and thing-driven operating system is required in the IoE. In this paper, we propose Sofie, which is a smart operating system for the IoE. As shown in **Fig. 2**, Sofie is sitting between devices and services, as both a service provider of the upper layer and a hardware manager for underlying devices, to provide high quality data through well performed things. For the IoE, Sofie is the brain that manages data, devices, and services. For service practitioners, Sofie is capable of reducing the complexity of development by offering an abstracted data access interface. In regard to functionality, Sofie is divided into two layers: the things management and data management.

The general architecture design is shown in **Fig. 3**. On the one hand, Sofie has the capability of maintaining all the connected things, since most of the involved things just work passively to generate data, while their number is too large to be tracked manually. On the other hand, Sofie is designed to encapsulate underlying things well, and only provide the data access interface, since super-stratum services only care about the data and do not need to know the low-level status. In the remaining part of this session, we will introduce Sofie from the views of things and data respectively.

2.1 Things Management

Taking control of different kinds of devices and sensors, Sofie has responsibility to reduce human intervention work, and coordinate all the devices to guarantee stable functionality and performance. In order to support qualified performance, the things management layer is made up of two parts: things config-

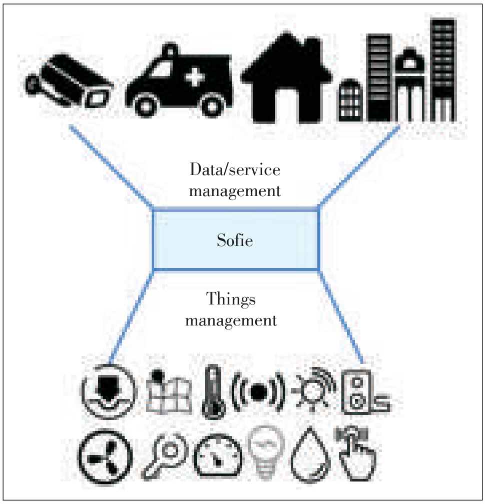
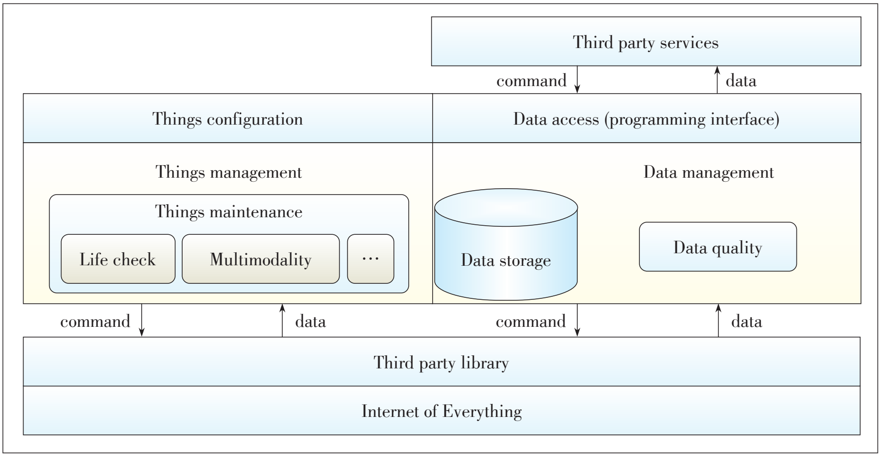


Figure 2.▶ Overview of Sofie.



▲**Figure 3. Architecture of Sofie.**

uration and things maintenance.

2.1.1 Things Configuration

Things configuration includes things registration and replacement. Although manual operation is inevitable in this part, Sofie takes care of all the remaining work after the device is connected to the system. When a new device is connected to the system, Sofie searches configuration files for its available service. Meanwhile, the identity information of a new device is recorded for Sofie to distinguish the new added device from other devices.

If a device dies or does not perform well, Sofie would raise a replacement request. Before the replacement happens, a compensation method is needed to preserve the stable performance and avoid any service interruption or damage to the system. In a relatively small and closed environment, suspending all services related to the malfunctioning device is a practicable method. In a relatively large and open environment, where services are more complex and mature, one alternative way is assigning the tasks of the malfunctioned device to adjacent devices temporarily. After the replacement is complete, Sofie will restore the states of the malfunctioning device to the new one, including the related services or functions.

2.1.2 Things Maintenance

Given the complexity and specificity of the IoE, it is not feasible for a human to take care of all the devices [12]-[16]. Thus, things maintenance is actually the mechanism of self-management for Sofie. When there is no human intervention, Sofie is the dominator to coordinate all the devices and sensors.

There are several ways to monitor health status of things. In this paper, we propose the following two techniques that will be implemented in Section 5.The general method is life checking. For each device connected to Sofie, a heartbeat signal is required to be sent to the system in regular time intervals. If no heartbeat is received from a certain device, Sofie could figure out the device is dead or disconnected from the system, and a relevant compensation method could be raised. In addition to life checking, which is seen as the regular monitoring method, some unconventional monitoring ways would be very helpful in extraordinary situations. For instance, when a device keeps sending heartbeats, while sending irregular data to the system compared to previous records, it can be reasonably inferred that there is something wrong with that device. Multimodality checking is another way to monitor the health status of things in Sofie. It harmoniously combines the different capabilities of connected devices as an auxiliary means to check the possible failure of a certain device. Smart home is one scenario that could implement this method. Data could be used here including video/image information recorded by cameras, temperature information recorded by temperature sensors, sound information recorded by acoustic sensors, etc. Correspondingly, multimodality checking could be adopted between a camera and a light, an audio system and an acoustic sensor, an air conditioning system and a temperature sensor and so on. To be specific, if a light component is not operating normally, Sofie could invoke domestic security camera to check if the light bulb works when its service is switched to on mode. The camera could take a picture or a short video of the surrounding environment of the light, and send it to the system for analysis. Similar to life checking process, if a device is observed to perform poorly, relevant remedial actions will be initiated. These actions could call neighboring devices to temporarily share the task of the broken device, or ask for a replacement.

2.2 Data Management

As a things-driven and data-driven operating system, data is another criterion to Sofie. Data management contains not only data quality, but also data storage and data access.

2.2.1 Data Quality

As we discussed before, the IoE is a broad concept, including different kinds of hardware that anyone can think of. Things in it could be as small as a motion sensor, or as large as a city. Due to the limitation of computational capacity, energy capacity and so on, most of the things only produce data in a passive way. Due to unstable wireless connection and volatile environment, the IoE is fragile to some extent. Considering the unreliable environment of the IoE, it is important that the data is of high quality, which means the data should be valid, complete, and timely. In general, the data quality of Sofie could be evaluated by two criteria: history pattern and reference data.

Data easily form a certain pattern due to the periodical activity of human and nature. To detect abnormal data and provide high quality data, historical data records could be employed by some data mining algorithms. If newly acquired data differ significantly from the history pattern, Sofie would analyze the reason for that, which could be device failure, communication interface change, or hostile attack.

In some cases, it is unavoidable that some devices or sensors malfunction. For example, in the case of several traffic cameras over road junctions where one traffic camera malfunctions and keeps sending low quality data, the data processed by nearby cameras could be leveraged as the reference before a replacement is complete.

2.2.2 Data Storage

Data storage is integral to Sofie data management. Underlying devices of Sofie generate a large variety of data every day. Some of the data is just redundant, while some could be valuable for future use. To cater the design that service should be isolated from things, valuable data should be abstracted before being stored in the database.

The database in Sofie is used to record the following four kinds of information:

- Device information. When a new device is added to Sofie, the identity information is recorded, as well as some con- figuration related information, such as related services. When a replacement happens, this kind of information could be used to restore previous configuration.
- State history. Basically, the state of a device could be divided into three cases: on, off, and unreachable. Both on and off indicate the device is available to the system, while unreachable state shows the device is disconnected. State information is significant for Sofie to manage things.
- Event history. It records every action of a device triggered by a related service. Event history would be useful when analyzing the history pattern of a certain device, or retrieving an event at a given time.
- File path. This is an alternative information, specifically designed for systems that include cameras. For such systems, video or image storage usually consumes a significant amount of space. Therefore, storing file paths in database is meaningful for secondary development.

2.2.3 Data Access

Without a uniform programming interface, developers would spend considerable time and effort to get data from different devices. Created for upper layer service, a programming interface is developed to make data access in the IoE flexible and convenient to service practitioners, as well as a tool against potential malicious attacks. Utilizing the database, the developer is able to utilize the unified interface to get abstracted data from Sofie.

3 Sofie at Home

In this section, we present the home environment as an example for Sofie, show how to design and implement a home OS based on Sofie, and address the widespread latency challenge in smart home environment.

The design of Sofie at home is shown in **Fig. 4**, including seven components: the communication adapter, event hub, database, self-learning engine, application programming interface, service registry, and name management that stretches across other components.

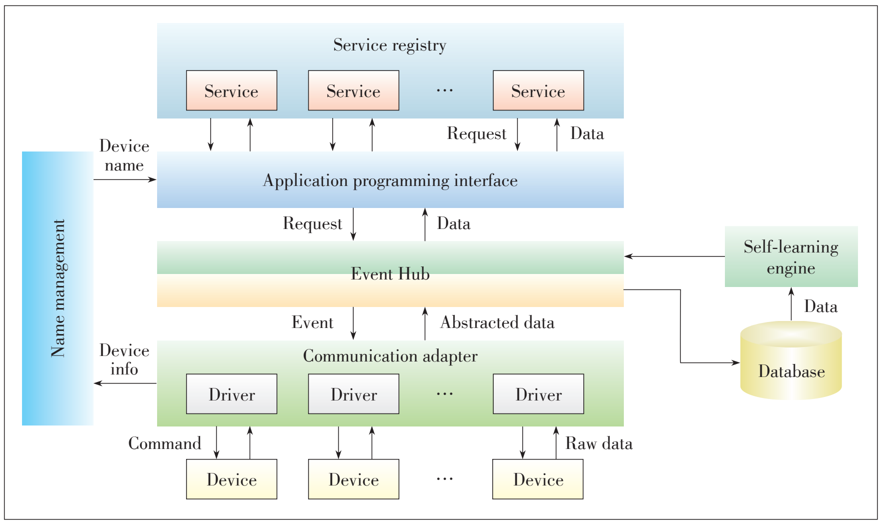
To integrate a device into Sofie, the communication adapter gets access to the device by the embedded drivers. These drivers are responsible for sending commands to devices and collecting state data (raw data) from them. Sitting between devices and the event hub, the communication adapter packages different communication methods that come from various kind of devices, while providing a uniform interface for upper layers' invocation. In this way, developers and users do not need to deal with multiple kinds of communication methods when manipulating the system. Moreover, it only provides abstracted data to upper layer components, reducing privacy risk to some extent.

As the core of the architecture, the event hub maps to two layers in the logical view: the data management and self-management layers. The event hub is responsible for capturing system events and sending instructions to lower levels. Those instructions are smart commands based on machine learning developed through communication with the self-learning engine. It collects requests from services and sends them to the communication adapter, and in turn, collects abstracted data from the communication adapter and sends them to upper layers. The database is another component in the data management layer. As a data-oriented system, Sofie generates large amount of data every day, which contains valuable information related to user preferences and settings. The event hub stores data in the database. The data stored in the database is utilized by the self-learning engine that belongs to the self-management layer. The self-learning engine creates a learning model. This learning model called the self-learning model acts as an input to the event hub to provide decision-making capability. To provide better user experience, the self-learning engine is developed to analyze user behavior, generate the personal model for the user, and help improve the system.

The application programming interface (API) and service registry are located in the upper layers of the system, and are utilized by third-party services. Developers are encouraged to use Sofie APIs to communicate with the event hub, and register their services with the system.

Required by all the layers, the name management module helps the system keep devices organized. When a new device is registered with the system, this module allocates a name for it using the following rule: location (where), role (who), and data description (what). This rule is complied by all the layers.

The design of Sofie is quite flexible to accommodate multiple functionality. Recently, with the cognitive technique development, there is a growing market for voice-controllable smart home products like Amazon Echo [17], Google Home [18], and



▲Figure 4. Design of Sofie at home.

Apple HomePod [19]. Such audio-based function is able to be realized in Sofie as a registered service, with the presence of suitable Natural Language Processing (NLP) model [20].

4 Smart Home Systems

There are many systems that can be used to build a smart home. We did not build Sofie from scratch but tried to utilize available systems to fulfill our concepts. In this section, we will review both commercial and open source products for smart homes.

4.1 Commercial Products

With the proliferation of the high-speed Internet and IoE, more and more products for the smart home are also available on the market. A smart device such as iRobot, Philips Hue, and Nest learning thermostat shows that homeowners are ready to embrace smart devices in their daily lives. Amazon Echo, Samsung SmartThings, and Google Home provide a hub and user interface for occupants to interact with connected devices. HomeOS from Microsoft and HomeKit from Apple enable a framework for communicating with and controlling connected accessories in a smart home.

Although there are a bunch of choices on the market to implement a smart home, the lack of proprietary software APIs and competitors' product support makes it hard to use commercial smart home systems in our project.

4.2 Open Source Systems

Despite the commercial products on the market, there are several communities that worked on open source projects for home automation systems. In this section, we compare the popular open source systems that can be used in a smart home, and hope this information could be helpful for the practitioners who plan to leverage open source home automation system in their own work.

In **Table 1**, we compared six open source systems from various aspects including programming languages and documentation support. Different object-oriented programming languages are used in these systems such as Python, Java, C++, C#, and Perl. We also inspected the lines of code for each system. We found out that all the systems' lines of source code fall into the similar magnitude between 100,000 to 1,000,000. Most of the systems except MisterHouse [21] use HTML to provide an interface to interact with the end user on the front end. Some of the systems also developed front end native mobile applications for iOS and Android such as Home Assistant [22], openHAB [23], and HomeGenie [24]. Becides MisterHouse, all of the other systems offer an API for the practitioners to utilize their data and functions. For the data storage, Home Assistant, Domoticz [25], and HomeGenie use SQLite database. OpenHAB and Freedomotic [26] provide data persistence service to the user, which means the user can freely implement customized database through the offered interface. In MisterHouse, the history data is not stored, and the users need to code directly into the source file if they want to implement automation methods. Meanwhile in all of the other systems, automation can be implemented through scripts file. Moreover, for Home Assistant, openHAB, and Freedomotic, automation could also be easily implemented by offering "trigger-condition-action" rules. All of the systems can be running on multiple platforms such as Linux, Windows, and Mac OS.

In the previous sections, we talked about our design of device abstraction and data abstraction. During the review, we found out that none of the current open source systems offered

▼Table 1. Comparison of smart home systems

	Language	Lines of code	Frontend	API	Data storage	Automation	Platform	Device abstraction	Data abstraction	Documentation
Home Assistant	Python3	213,901	HTML, iOS	Y	SQLite	Rules, Scripts	Linux, Win, Mac OS X	Y	N	Good
openHAB	Java	904,316	HTML, Android, iOS, Win	Y	Persistence services	Rules, Scripts	Any device with JVM	Y	N	Good
Domoticz	C++	645,682	HTML	Y	SQLite	Scripts	Linux, Win, Mac OS X	N	N	Poor
Freedomotic	Java	159,976	HTML	Y	Data persistence	Rules, Scripts	Any device with JVM	N	N	Good
HomeGenie	C#	282,724	HTML, Android	Y	SQLite	Scripts	Linux, Win, Mac OS X	N	N	Average
MisterHouse	Perl	690,887	NA	N	NA	Perl code	Linux, Win, Mac OS X	N	N	Poor

API: application programming interface JVM: Java virtual machine OS: operating system data abstraction function, which means they only store the raw data reported by the devices. Only Home Assistant and openHAB implement device abstraction. They can isolate devices from services by abstracting the role of devices. In this way, services do not need to know the physical information of devices such as IP and MAC addresses.

Documentation is the last index we used to evaluate the systems. A well written and maintained document is extremely helpful for the practitioners to install, configure, and manage the system. Home Assistant, openHAB, and Freedomotic provide good documents for the users.

Based on our evaluation of the above open source systems, we decided to use Home Assistant and openHAB in our project since their architectures fit our design better. We eventually chose Home Assistant over openHAB since Python has less overhead than Java, and the Home Assistant community is more active with updates and latest device support.

5 Implementation

In this section, we will introduce the architecture of Home Assistant and the implementation of Sofie on top of it.

5.1 Architecture of Home Assistant

Fig. 5 shows the architecture of Home Assistant. For the things with open APIs (as most of the products on the market are), there are usually some third party libraries written in Python by the community. Home Assistant utilizes those libraries and abstract things into different components. Then on top of the components is the core of Home Assistant, where a state machine, an event bus, and a service registry are supported to control and manage the components. A user interface is offered on top of the Home Assistant core for users to access the home information and control devices. Home automation is also supported by providing a YAML (YAML Ain't Markup Language) [27] configuration file.

5.2 Sofie on Top of Home Assistant

Based on the comparison of open source smart home systems, we chose the communication layer of Home Assistant as the substructure, and developed Sofie on top of it. As we claimed in the previous section, Home Assistant has implemented device abstraction in a comprehensive way. More than 600 components are supported by Home Assistant, and universal interfaces are provided for different kinds of devices. We use Home Assistant as an open source package to build Sofie prototype because Home Assistant solves the driver issue, which facilitates the creation of Sofie, and avoids redundant work.

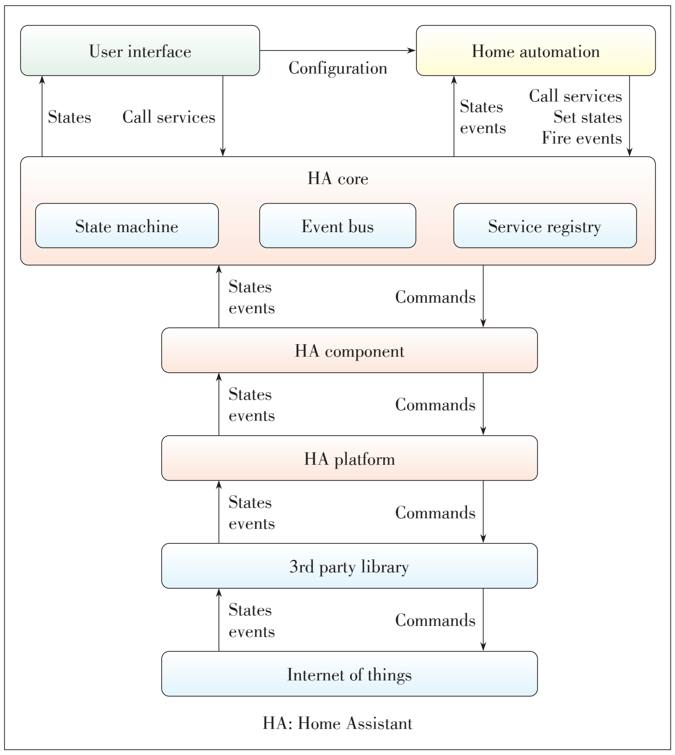
To successfully implement functions required by Sofie, we modified the components in the event bus layer and service layer, as shown in **Fig. 6**. In comparison with Home Assistant, which focuses on the functionality of the system, Sofie pays attention to both functionality and the self-management of devices. To demonstrate Sofie's capability of data management and things management, we conducted several preliminary experiments by leveraging Home Assistant framework.

5.2.1 Database Renovation

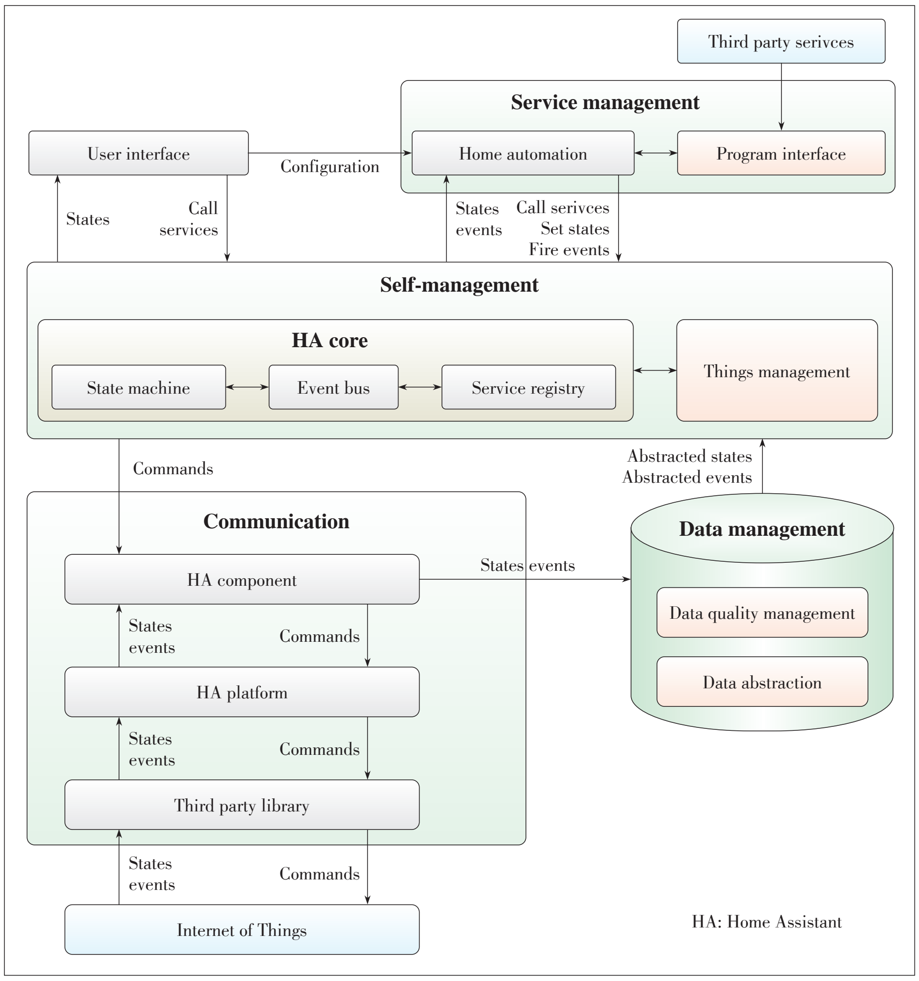
Home Assistant uses SQLite as the default back end database to store historical data. There are four tables in the database:

- The events table records what type of event happens of what device and at what time. It is frequently updated once a service is registered, a new platform is connected, or a device state is changed, and the detailed information will be stored in this table.
- The recorder runs table records the start and end time of the whole system. It only updates when Home Assistant starts or ends.
- The schema changes table records the system update history. It seldom receives and stores new data compared to other tables.
- The states table records changes in device status, including the last changed time and last updated time. It is an active table since the device state changes from time to time in the smart home environment.

To successfully realize the functionality of Sofie, we added the following two tables. One is the IP address table to store the IP addresses of registered devices. It updates when new devices are added to the system, when a replacement occurs, or



▲Figure 5. Architecture of Home Assistant.



▲Figure 6. Architecture of Sofie on top of Home Assistant.

when a connected device changes the IP address. The life checking component relies on this table to check the reachability of registered devices. The other is the file path table to store the image/video files recorded by security cameras. Video images are important information for both Sofie and top layer services. For Sofie, this information could be utilized in device management. More details will be discussed in 5.2.4. Services built on top of Sofie may need to fetch historical image data to finish the tasks. For instance, this table could provide the path of related files to a service by analyzing historical data when the service needs to track a specific person or an event. The update frequency of this table depends on the setting and could vary greatly. In our experiment, it is only updated when a certain event occurs.

5.2.2 Life Checking

Based on the Sofie design discussed in Section 2, monitoring the healthy status of devices could help maintain system performance. The general method to track the devices is life checking. Considering the passive character of most devices, the core is set as the action initiator to start the life checking. In our configuration, all the IP addresses in the database are attempted in a round-robin manner by the system every five minutes. If an IP address could not be accessed, Sofie will suspend its current configuration and mark the state as “unreachable” both in database and user interface. Otherwise, the working state will be displayed on the user interface. Generally speaking, “unreachable” is not necessary because of device failure just because the user turned off the device through its physical switch, or due to environment power outage.

Fig. 7 is an example of a user interface with life checking function. There are four devices in the living room: a table lamp controlled by a smart switch, ceiling lights composed of several smart light bulbs, a smart thermostat, and Chromecast. During the life checking round, the Chromecast is inaccessible, and therefore the user interface displays “unreachable” as its current state.

5.2.3 APIs

Sofie provides two APIs for practitioners to work with: a Python API inherited from Home Assistant to interact with things, and a database application programming interface (DB-API) for SQLite database. With the Python API shown in **Fig. 8**, developers can connect to Sofie remotely with IP address and password combination, get data from Sofie, and also control the devices. With DB-API shown in **Fig. 9**, developers can connect to the Sofie database directly if the service only cares about the home data.

5.2.4 Multimodality Checking

Another way to check the healthy status of devices is multi-

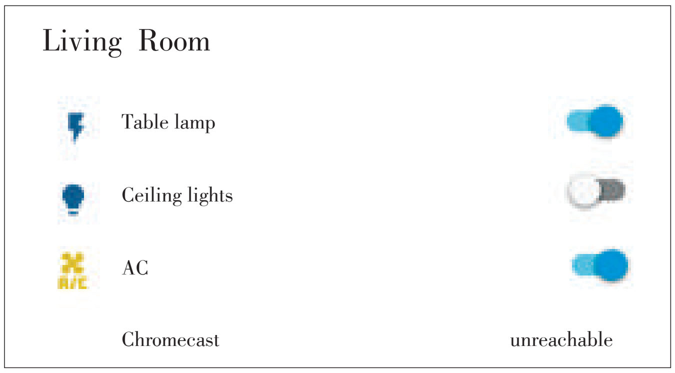

Living Room

Table lamp	on
Ceiling lights	off
AC	on
Chromecast	unreachable

▲Figure 7. Home Assistant user interface with life checking.

```
# Python-API interface for Sofie inherited from HA
import homeassistant.remote as remote

# Connect to Sofie Hub
api = remote.API('192.168.0.1', 'YOUR_PASSWORD')

# Get available services, events, and entities
services = remote.get_services(api)
events = remote.get_event_listeners(api)
entities = remote.get_states(api)
```

Figure 8.
Python API example code for Sofie.

```
# DB-API interface for SQLite databases
import sqlite3

# Connect to database
comm = sqlite3.connect('Sofie.db')
c = conn.cursor()

# Query data from database
c.execute("SELECT * FROM event_data WHERE
entity_id='switch.livingronn_switch'")

# Save (commit) the changes
conn.comit()

# Close the connection
conn.close()
```

Figure 9.
DB-API example code for Sofie.

modality checking. When a new device is registered, the user is allowed to set up a rule for multimodality checking. The rule manner is *Name_of_device1: multimodal: name_of_device2*. This rule states that Device 2 will be invoked when Sofie checks the healthy status of Device 1. Such status check relationship could exist in different kinds of devices such as light and camera, AC system and temperature sensor, and sound system and acoustic sensor. In the preliminary experiment, we used light and camera as an example, in which the security camera plays a key role in taking pictures and utilizing vision related libraries to process the captured images. Assuming a relatively dark background, Sofie checks if the light is operating normally by triggering the camera to take a picture when the light turns on and compare it with a previously saved one when the light was off. Then Sofie retrieves the vision related library to compare the difference between the two images to check if the light successfully turned on.

Figs. 10 and **11** show the multimodality checking process. The pictures in Figs. 10a and 10b are taken from the same angle, and the same applies to Figs. 10c and 10d. The generated gray level histogram of each picture is shown in Fig. 11. Figs. 11c and 11f are the gray level histogram of difference image of Figs. 10a and 10b, and Figs. 10c and 10d respectively. To analyze the grey level images, an appropriate threshold method is meaningful [28]. As a preliminary experiment, we set the threshold of pixel gray level to 50. That is, if more than half of the pixels are larger than 50 in the image difference gray level histogram, the system treats the light as turned on. The same method standard can also be used to confirm the light is turned

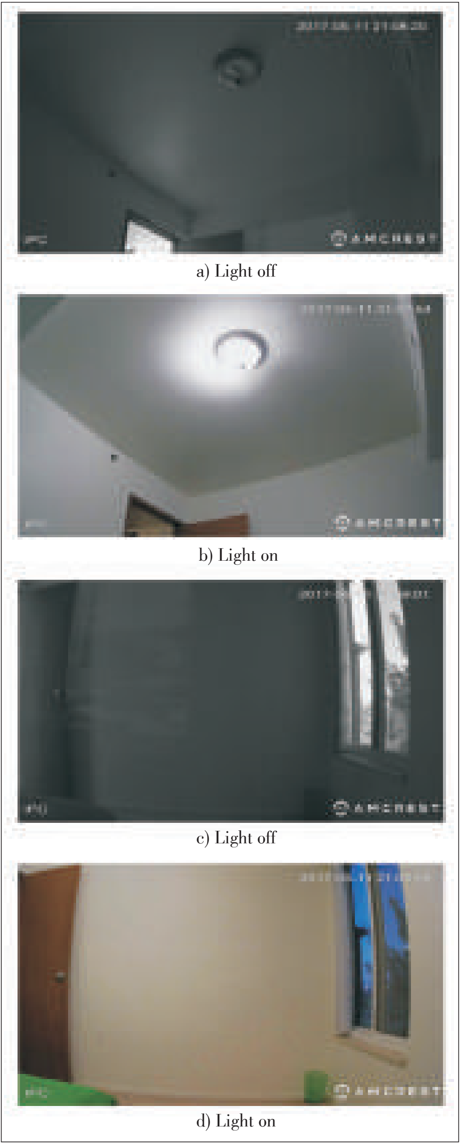
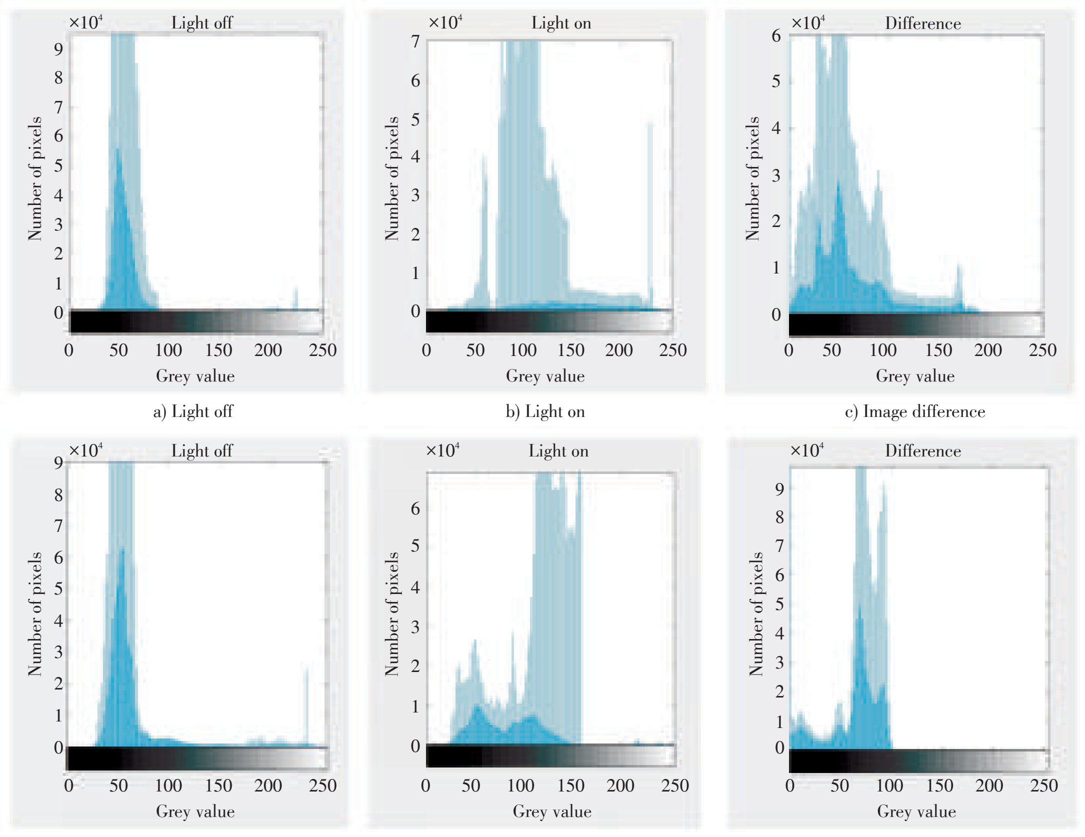


Figure 10.
Camera snapshots when light turns on/off.

off. During the preliminary experiment, the average response time is 220 ms for a complete multimodality checking process, which including taking picture, transferring message, comparing two pictures, and getting the conclusion. Multimodality checking is not limited to monitoring a light's state. In a mature smart home running Sofie, the security camera has the ability to check several devices such as a smart stove and a washer machine.

5.2.5 Configuration File

When Sofie starts, it will reach out to the configuration file to set up the initial smart home environment. Inherited from Home Assistant, the configuration file is consisted of two parts: 1) default settings, including the user interface (username and password) and geographic location (longitude, latitude, time zone, etc.); 2) custom settings input by the user, telling the system what kind of device is ready to connect and what kind of



a) Light off b) Light on c) Image difference

d) Light off e) Light on f) Image difference

Figure 11. Gray level histogram corresponding to the snapshots in Fig. 10.

service is required. A sample configuration file is shown in **Fig. 12**. Usually, same kind of devices are grouped to the same component, then distinguished by different platforms (brands) and IP addresses. Fig. 12b defines one media player (Roku TV), one light group (control through Phillips Hue), and two security cameras (Amcrest camera, and MJPEG camera). According to this configuration file, the living room camera is involved in the multimodality checking of light.

In this section, we introduced how we implement Sofie for a smart home, and we also showed how to implement our design on the top of an open source system. An open source package of Sofie is going to be released on our website soon.

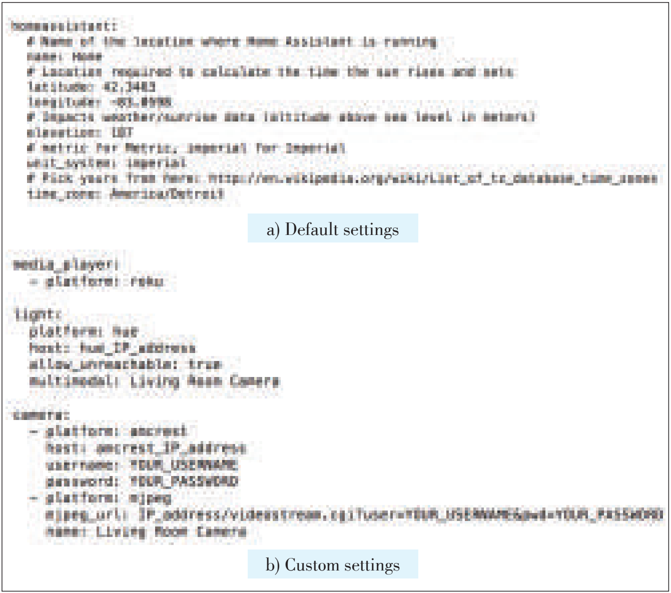
6 Discussion

Here, we share some lessons and experience we have learned during the design and development of a smart home.

6.1 Latency in Sofie at Home

In a smart home, the response time is an important metric to determine whether a smart home system is satisfactory or not [29]–[35]. Sofie is no exception. Back in 1968, Miller described computer mainframe responsiveness in three different orders of magnitude [36]: 1) 100 ms is perceived as instantaneous; 2) 1 s or less is fast enough for the user to interact with the system in a free way; 3) 10 s or more reduces the user's interest. Generally speaking, controlling the response time under 1 s is sufficient for the satisfactory functionality of the smart home implementation, and meeting user expectations.

Fig. 13 shows the latency of turning on light in both conven-



```
homeassistant:
  # Name of the location where Home Assistant is running
  name: Home
  # Location required to calculate the time the sun rises and sets
  latitude: 42.3483
  longitude: -83.0998
  # Impacts weather/sunrise data (altitude above sea level in meters)
  elevation: 187
  # metric for Metric, imperial for Imperial
  unit_system: imperial
  # Pick yours from here: http://en.wikipedia.org/wiki/List_of_tz_database_time_zones
  time_zone: America/Detroit
```

a) Default settings

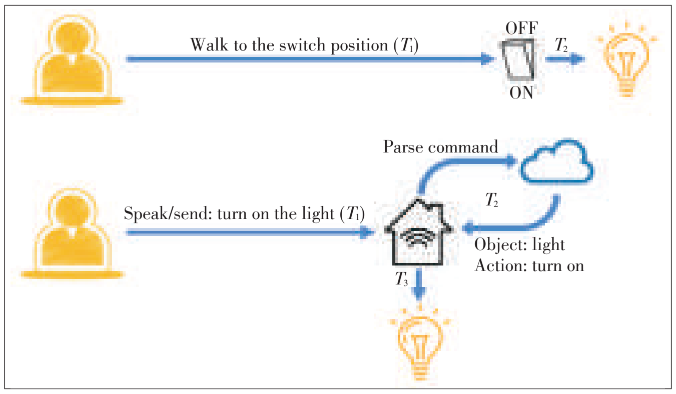
```
media_player:
  - platform: roku

light:
  platform: hue
  host: hue_IP_address
  allow_unreachable: true
  multimodal: Living Room Camera

camera:
  - platform: amcrest
    host: amcrest_IP_address
    username: YOUR_USERNAME
    password: YOUR_PASSWORD
  - platform: mjpeg
    mjpeg_url: IP_address/videostream.cgi?user=YOUR_USERNAME&pwd=YOUR_PASSWORD
    name: Living Room Camera
```

b) Custom settings

Figure 12. Configuration file example code for Sofie.


Walk to the switch position (T_1)
OFF
ON
T_2
Parse command
T_2
Speak/send: turn on the light (T_1)
Object: light
Action: turn on
T_3

▲Figure 13. Latency in turning on light.

tional and smart home scenarios. When a user walks to a physical switch position to turn on the light (Fig. 13a), we assume T_1 is the time that takes a user to physically walk to the light switch position, and T_2 is the time to switch the light from off to on position, then the total response time is $T_1+T_2 \approx T_1$ since T_2 is perceived as instantaneous ($T_2 \approx 0$). Therefore, the latency is T_1 in this scenario. On the other hand, in Sofie the response time for turning on the light consists of three parts (Fig. 13b): The time for user sending the command by voice or user interface (T_1), that for Sofie parsing the command to determine the object and action (T_2), and that for Sofie turning on the light according to the command (T_3). For the user, the expected response time is counted after he actuates the command ($T_1 \approx 0$). Thus, the response time is T_2+T_3. As discussed earlier, the response time should be limited to 1 s for the user to interact freely and easily with the system. That is $T_2+T_3 \leqslant 1$ s. During this process, many factors could potentially contribute to the latency, including software components, hardware configuration, and network conditions. In our preliminary experiment, turning on the light via a user interface takes less than 200 ms for an event from initiation to completion in a stable Wi-Fi environment. However, if the light is turned on by using a voice-controlled service, Sofie will face more challenges by spending more time in capturing and parsing voice commands [37], [38]. This is mainly due to dependence on NLP models. Thus, a well-trained NLP model and search engine are urgently required for a satisfactory response time in Sofie.

6.2 Lessons and Experience

Whether it is a technical or non-technical task, there is a minimum skill level required to complete the task on hand. Depending on the task complexity and the field the task belongs to, knowledge in more than one field or area might be required.

Converting a non-smart home to a smart one requires some knowledge in software, hardware, and networking. For a non-technical homeowner, the Do-It-Yourself (DIY) task of converting a non-smart home to a smart one can be difficult and overwhelming. It will take several trials and errors before such a task is completed successfully. As an example, a DIY wireless surveillance system project will require 1) hardware configuration, 2) software installation and configuration, and 3) maintenance. Therefore, there is a considerable amount of manual work involved that most homeowners will find cumbersome.

Open source home automation software like Home Assistant provides a solution for integrating various home automation systems into one single solution via a uniform user interface. Although such open source software is vendor- neutral, hardware/protocol-agnostic, extensible, and platform independent, a homeowner has to invest a considerable amount of time in learning its concept and architecture in order to setup a customized smart home. The steep learning curve revolves around the setup and configuration of the system. That is, the existing open source software is neither plug and play nor a commercial off-the-shelf (COTS) product.

In addition to that, the current open source software is still not a data-oriented system; therefore, it does not involve machine learning. To further elaborate on that point, an experienced smart home user can utilize external software by creating scripts to communicate with the open source software. As an example, a user can use software like Blueiris to consume open source software API in order to trigger events. As an example, Blueiris can communicate with Home Assistant to turn on lights, TV, and so on when motion is detected. That is, any device (thing) on the home automation bus can be controlled remotely when motion is detected.

7 Conclusions

In this paper, we proposed Sofie, a smart operating system for the IoE, and discussed its design and implementation. We illustrated the design of Sofie at home and the significance of latency in a smart home environment. We then compared open source and commercial smart home systems that are available on the market nowadays. When explaining the implementation of Sofie in detail, we attempted to discuss a few of IoE-associated challenges that are related to configuration and maintenance. In general, whether it is a smart home or a connected vehicle, the user experience is very critical to the success of IoE applications. If it is hard to configure, maintain, and communicate with devices, the smart home experience will not be successful. If it takes 30 seconds to turn on a light remotely, the chances are that the user would not use the application for long. We showed that Sofie could be helpful in the IoE to practitioners to better manage their things, data, and services.

References

[1] 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, Jun. 2016. doi: 10.1109/JIOT.2016.2579198.
[2] Cisco, "Cisco global cloud index: forecast and methodology, 2014-2019," Cisco, white paper, 2014.
[3] D. Evans, "The internet of things: how the next evolution of the internet is changing everything," CISCO white paper, vol. 1, p. 14, 2011.

[4] Y. Strengers. (2016, Jun. 10). *Creating pleasance: new needs for the smart home* [Online]. Available: http://www.demand.ac.uk/10/06/2016/creating-pleasance-new-needs-for-the-smart-home-yolande-strengers
[5] D.-L. Wang, "The internet of things the design and implementation of smart home control system," in *IEEE International Conference on Robots & Intelligent System (ICRIS)*, Zhangjiajie, China, Dec. 2016, pp. 449-452. doi: 10.1109/ICRIS.2016.95.
[6] U. Bakar, H. Ghayvat, S. Hasanm, and S. Mukhopadhyay, "Activity and anomaly detection in smart home: a survey," in *Next Generation Sensors and Systems*. Berlin/Heidelberg, Germany: Springer, 2016, pp. 191–220.
[7] Y. Strengers, "Envisioning the smart home: reimagining a smart energy future1," in *Digital Materialities: Design and Anthropology*, S. Pink, E. Ardevol, and D. Lanzeni ed. London, UK: Bloomsbury Publishing, 2016.
[8] E. Ahmed, I. Yaqoob, A. Gani, M. Imran, and M. Guizani, "Internet-of-things-based smart environments: state of the art, taxonomy, and open research challenges," *IEEE Wireless Communications*, vol. 23, no. 5, pp. 10-16, Nov. 2016. doi: 10.1109/MWC.2016.7721736.
[9] N. Jiang, C. Schmidt, V. Matossian, and M. Parashar, "Enabling applications in sensor-based pervasive environments," in *Proc. 1st Workshop on Broadband Advanced Sensor Networks (BaseNets)*, San Jose, USA, 2004, p. 48.
[10] M. Gowda, A. Dhekne, S. Shen, et al., "Bringing iot to sports analytics," in *14th USENIX Symposium on Networked Systems Design and Implementation*, Boston, USA, 2017, pp. 498–513.
[11] D. Vasisht, Z. Kapetanovic, J. Won, et al., "Farmbeats: An iot platform for data-driven agriculture," in *14th USENIX Symposium on Networked Systems Design and Implementation*, Boston, USA, 2017, pp. 514–529.
[12] E. Soltanaghaei and K. Whitehouse, "Walksense: Classifying home occupancy states using walkway sensing," in *Proc. 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*, Stanford, USA, 2016, pp. 167–176.
[13] Y. Agarwal, B. Balaji, R. Gupta, et al., "Occupancy-driven energy management for smart building automation," in *Proc. 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*, Zurich, Switzerland, 2010, pp. 1–6. doi: 10.1145/1878431.1878433.
[14] D. Austin, Z. T. Beattie, T. Riley, et al., "Unobtrusive classification of sleep and wakefulness using load cells under the bed," in *Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, San Diego, USA, 2012, pp. 5254–5257. doi: 10.1109/EMBC.2012.6347179.
[15] G. Gao and K. Whitehouse, "The self-programming thermostat: optimizing setback schedules based on home occupancy patterns," in *Proc. First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*, Berkeley, California, 2009, pp. 67–72. doi: 10.1145/1810279.1810294.
[16] G. Zhang and M. Parashar, "Context-aware dynamic access control for pervasive applications," in *Pro. Communication Networks and Distributed Systems Modeling and Simulation Conference*, San Diego, USA, 2004, pp. 21–30.
[17] Amazon. (2017, Jun. 2). Amazon echo [Online]. Available: https://www.amazon.com/Amazon-Echo-Bluetooth-Speaker- with-WiFi-Alexa/dp/B00X4WHP5E
[18] Google. (2017, Jun. 2). Google home [Online]. Available: https://madeby.google.com/home
[19] Apple. (2017, Jun. 6). Apple homepod [Online]. Available: https://www.apple.com/homepod
[20] M. Chandak and R. Dharaskar. (2010, Apr.). Natural language processing based context sensitive, content specific architecture & its speech based implementation for smart home applications. *International Journal of Smart Home* [Online]. *4(2)*. Available: http://www.sersc.org/journals/IJSH/vol4_no2_2010/1.pdf
[21] MisterHouse. (2017, May 10). MisterHouse—it knows kung-fu [Online]. Available: http://misterhouse.sourceforge.net
[22] Home Assistant. (2017, May 10). Home assistant, an open-source home automation platform running on python 3 [Online]. Available: https://home-assistant.io
[23] OpenHAB. (2017, May 10). OpenHAB, a vendor and technology agnostic open source automation sofware for your home [Online]. Available: https://www.openhab.org
[24] HomeGenie. (2017, May 10). HomeGenie, the open source, programmable, home automation server for smart connected devices and applications [Online]. Available: http://www.homegenie.it
[25] Domoticz. (2017, May 10). Domoticz, control at your finger tips [Online]. Available: http://www.domoticz.com
[26] Freedomotic. (2017, May 10). Freedomotic, open IoT framework [Online]. Available: http://www.freedomotic.com
[27] YAML. (2017, May 10). Yaml ain't markup language [Online]. Available: http://yaml.org
[28] M. Sezgin *et al.*, "Survey over image thresholding techniques and quantitative performance evaluation," *Journal of Electronic imaging*, vol. 13, no. 1, pp. 146–168, Jan. 2004.
[29] Safehome. (2017, Jun. 2). Best home security company response times [Online]. Available: https://www.safehome.org/security- systems/best/response-times
[30] K. M. Tsui and S.-C. Chan, "Demand response optimization for smart home scheduling under real-time pricing," *IEEE Transactions on Smart Grid*, vol. 3, no. 4, pp. 1812–1821, Dec. 2012. doi: 10.1109/TSG.2012.2218835.
[31] F. Fernandes, H. Morais, Z. Vale, and C. Ramos, "Dynamic load management in a smart home to participate in demand response events," *Energy and Buildings*, vol. 82, pp. 592–606, Oct. 2014. doi: 10.1016/j.enbuild.2014.07.067.
[32] T.-Y. Chung, I. Mashal, O. Alsaryrah, et al., "Design and implementation of light-weight smart home gateway for social web of things," in *IEEE Sixth International Conf on Ubiquitous and Future Networks (ICUFN)*, Shanghai, China, Jul. 2014, pp. 425–430. doi: 10.1109/ICUFN.2014.6876827.
[33] M. Li and H.-J. Lin, "Design and implementation of smart home control systems based on wireless sensor networks and power line communications," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 7, pp. 4430–4442, Jul. 2015. doi: 10.1109/TIE.2014.2379586.
[34] Y. Ozturk, D. Senthilkumar, S. Kumar, and G. Lee, "An intelligent home energy management system to improve demand response," *IEEE Transactions on Smart Grid*, vol. 4, no. 2, pp. 694–701, Jun. 2013. doi: 10.1109/TSG.2012.2235088.
[35] R. Deng, Z. Yang, F. Hou, M.-Y. Chow, and J. Chen, "Distributed real-time demand response in multiseller–multibuyer smart distribution grid," *IEEE Transactions on Power Systems*, vol. 30, no. 5, pp. 2364- 2374, Sept. 2015. doi: 10.1109/TPWRS.2014.2359457.
[36] R. B. Miller, "Response time in man-computer conversational transactions," in *ACM AFIPS' 68*, San Francisco, USA, Dec.1968, pp. 267–277. doi: 10.1145/1476589.1476628.
[37] P. Chahuara, F. Portet, and M. Vacher, "Context-aware decision making under uncertainty for voice-based control of smart home," *Expert Systems with Applications*, vol. 75, pp. 63–79, Jun. 2017. doi: 10.1016/j.eswa.2017.01.014.
[38] M. R. Abid, E. M. Petriu, and E. Amjadian, "Dynamic sign language recognition for smart home interactive application using stochastic linear formal grammar," *IEEE Transactions on Instrumentation and Measurement*, vol. 64, no. 3, pp. 596–605, Sept. 2015. doi: 10.1109/TIM.2014.2351331.

Manuscript received: 2017-06-07

Biographies

CAO Jie (jiecao@wayne.edu) received his B.S. in telecommunication engineering from Xidian University, China and M.S. in computer science from Wayne State University, USA. He is currently pursuing his Ph.D. in computer science at Wayne State University and internship at Interdigital Inc. His research interests include edge computing, computer systems, and wireless health. He has published 5 research papers and his publication of MyPalmVein received the Best Student Paper Award from HealthCom, 2015.

XU Lanyu (xu.lanyu@wayne.edu) received her B.S. in electronic and information engineering from Tongji University, China. She is currently a Ph.D. candidate in computer science at Wayne State University, USA. Her research interests include edge computing, computer systems, and cognitive service.

Raef Abdallah (raef.abdallah@gmail.com) received his B.S. in computer science from Lebanese American University, Lebanon. He holds M.S. degrees in computer science and industrial engineering from Oklahoma State University, USA. His research interests include IoT, smart homes, simulation, and design of algorithms. He has developed solutions for major companies in the United States in the areas of education, manufacturing, and defense. He is currently working in the connected vehicle technology.

SHI Weisong (weisong@wayne.edu) is a Charles H. Gershenson Distinguished Faculty Fellow and a professor of Computer Science at Wayne State University, USA. His research interests include edge computing, computer systems, energy-efficiency, and wireless health. He received his B.S. from Xidian University, China in 1995, and Ph.D. from Chinese Academy of Sciences, China in 2000, both in computer engineering. He is a recipient of National Outstanding Ph.D. Dissertation Award of China and the NSF CAREER award. He is an IEEE Fellow and an ACM Distinguished Scientist.

DOI:10.3969/j.issn.1673-5188.2017.04.003
http://kns.cnki.net/kcms/detail/34.1294.TN.20171108.1011.002.html, published online November 08, 2017

HCOS: A Unified Model and Architecture for Cloud Operating System

CHEN Aiguo, WU Huaigu, TIAN Ling, and LUO Guangchun
(University of Electronic Science and Technology of China, Chengdu 611731, China)

Abstract

Currently, Infrastructure as a Service(IaaS) and Platform as a Service(PaaS) platforms play a role as a cloud operating system(COS). They are separated from each other in resource management, which may cause inconsistent resource status and result in the decrease of performance. Moreover, heterogeneous resources are not managed well in existing cloud computing platforms. Referring to the theory of operating system, we propose a unified architecture model of cloud operating system, which has six layers corresponding to the layered architecture of legacy operating system. Based on this architecture, a new cloud operating system called Hua-Cloud Computing System(HCOS) is realized. In HCOS, the hybrid resources are managed in a unified way. This method improves the unified scheduling capability of heterogeneous resources and eliminates the problem of resource status inconsistency. The main characteristics of HCOS are introduced and two typical applications are illustrated in this paper.

Keywords

cloud computing; IaaS; PaaS; cloud operating system

1 Introduction

Cloud computing is an emerging business computing model. It is seen as the third IT revolution following the personal computer revolution and the Internet revolution. It brings a fundamental change in lifestyle, production methods and business models. It integrates or segments physical resources into resource pools by using virtualization technology. Then a variety of applications could use virtualized resources according to their actual demands. Classically, cloud computing systems provide three main categories of cloud computing services: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). The supporting system of the three types of cloud services is cloud computing platform, which is usually built above large data centers.

The core software of cloud computing platform is cloud operating system (COS), which is also called cloud platform. COS is a set of control systems for applications running in the cloud environment. It works on hardware and legacy operating system, manages and schedules all resources, provides fundamental resource services and operation environments for cloud applications. From a technology perspective, COS defines a set of standard interfaces between cloud resources and cloud applications, ensuring the efficient and standardized use of resources. COS is also built with a set of efficient task and resource scheduling models, used to decide how to efficiently manage and allocate system resources.

Existing cloud operating system is divided into two parts: PaaS and IaaS. The two parts are completely independent and could be deployed or used respectively. The IaaS manages computing, storage and networking resources and provides basic resource services to the PaaS or to users directly. The PaaS supports the full life cycle management of cloud applications, including coding, testing, deployment and maintenance. It could be built on a traditional data center or on an IaaS-based cloud data center.

IaaS plus PaaS is a typical and complete cloud solution. For example, an architecture for cloud brokering given in [1] implements dependability properties in an end-to-end way involving different cloud actors and all over cloud service models including SaaS, PaaS and IaaS. However, in such systems with independent IaaS layer and PaaS layer, these two parts have their own message mechanisms and dependent scheduling models, which may cause some problems such as resource state inconsistency[2]. In addition, IaaS cannot perceive the global demand of the development or the task for high-level applications of PaaS layer, so it cannot make optimal scheduling operations.

This work was supported in part by Sichuan Provincial Public Security Bureau of China under Grant No. 2015TD0002, Unicom Cloud Data Co., Ltd of China under Grant No. 2013JQ0005, and Science and Technology Department of Sichuan Province under Grant no. 2017HH0075.

We call this as an IaaS and PaaS isolation problem (IPI problem for abbreviation). Moreover, there are usually massive heterogeneous resources in cloud computing environment. Currently, the unified management of heterogeneous resources (and operating systems) is not well supported in existing cloud operating systems.

IaaS and PaaS are becoming integrated [3]. In this context, aiming at solving the IPI problem and providing the unified management for heterogeneous resources, a new six-layered cloud operating system architecture is proposed in this paper, which is based on the theory of operating system. According to the architecture, a new unified Hua-Cloud Computing System (HCOS) is presented and described.

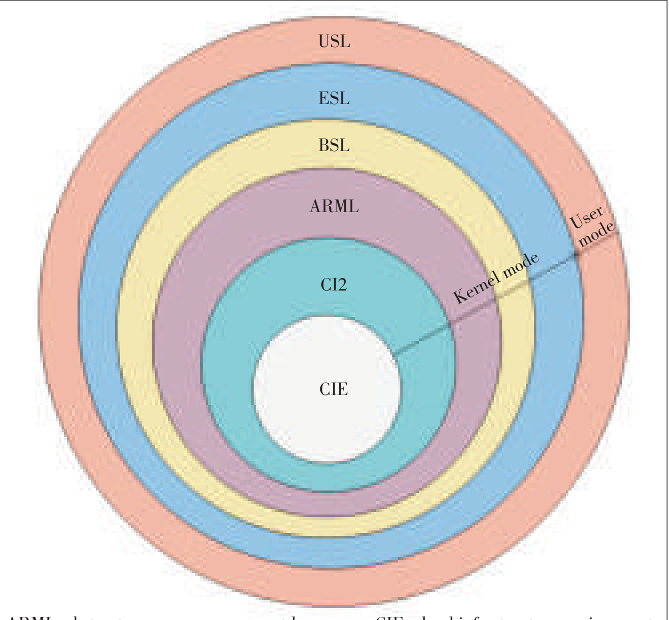
The remaining parts of this paper are organized as follows. Section 2 describes the architecture model, and the design of HCOS, Section 3 discusses the performance comparison and analysis of HCOS and a few existing cloud operating systems. The related work is discussed in Section 4 and the summary of our research work is given in Section 5.

2 Design and Implementation of HCOS

2.1 A Six-Layered Architecture Model of COS

According to the requirements of industrial application and the limitations of existing COS, we present a new unified cloud operating system architecture model referring to the theory of traditional operating system (**Fig. 1**).

As can be seen from Fig. 1, this cloud operating system from the inside to the outside has a total of six layers, and is divided into the kernel mode (from layer 1 to layer 5) and the user mode (layer 6). The working mechanism of this six-layer architecture is similar to the operating system (OS) of personal computer. It has distinctive features and advantages compared with the existing COS architecture. A cloud operating system designed and implemented in this six-layered architecture model can provide both IaaS and PaaS abilities in an integrated way.



ARML: abstract resource management layer
BSL: basic system layer
CI2: common infrastructure interface layer
CIE: cloud infrastructure environment
ESL: enterprise system layer
USL: user service layer

▲**Figure 1. Six-layered COS architecture model.**

The six layers are as follows.

1) Cloud Infrastructure Environment (CIE). It is the first layer and in the most central part within the six-layered model, which indicates the basic environment of cloud computing. It corresponds to the basis hardware of legacy OS. However, it does not belong to the cloud operating system itself. CIE is just used to define the raw infrastructure resource managed by the cloud operating system.
2) Common Infrastructure Interface Layer (CI2). The second layer is CI2, which corresponds to the driver layer of legacy OS. It provides drivers for resources of the infrastructure environment.
3) Abstract Resource Management Layer (ARML). The third layer is ARML, corresponding to the hardware abstraction layer of traditional OS. It achieves an abstract management of all resources defined in the first layer.
4) Basic System Layer (BSL). The fourth layer named BSL is the kernel of COS, corresponding to the logical resource layer (also known as the "kernel") of traditional OS. It supports applications of upper layers directly, and consists of OS installed computing systems, file systems and database systems (with prepared storage devices). Applications are running over this basic layer, similar to applications running over logic resource layer of traditional OS.
5) Enterprise System Layer (ESL). The fifth layer is ESL, which is also the function layer of cloud operating system, corresponding to the system call layer of traditional OS. The main function of this layer is providing business functions. By calling these functions, cloud applications can combine with basic system and then form complete business systems. The ESL also schedules business system in accordance with application requirements. Moreover, in order to adapt to the development trend of big data, it can provide special integrated data management and processing abilities.
6) User Service Layer (USL). The sixth layer is USL, corresponding to the library function and human-computer interaction layer of traditional OS. The main function of this layer is to provide a series of service interfaces, so that users can use cloud operating system conveniently. Through this layer, cloud operating system can be exposed the same abilities as traditional IaaS or PaaS platform. Obviously, cloud operating system can be used as an IaaS or a PaaS platform.

2.2 HCOS—A Implementation of Six-Layered Model

HCOS was designed and implemented according to the six-layered cloud operating system model. Its overall system architecture is shown in **Fig. 2**.

The six layers of HCOS are also CIE, CI2, ARML, BSL, ESL and USL. The basic resources of HCOS could come from public clouds, private clouds or hybrid clouds. ARML takes resources as standard computing, storage, networking, and software templates (including operating systems, database systems, etc.). BSL manages all the instances of the resources coming from ARML. ESL is composed of a number of basic system instances on demand. And USL provides a variety of business systems, resources and capabilities to the user as services. In addition, HCOS also includes Common Monitoring Logging Management (CML) and Unified Security Centre (USC). Above HCOS, we can provide different cloud services through corresponding portals, such IaaS portal, PaaS portal, and uniform cloud portal. The overall architecture of HCOS will be described in detail.

1) CIE

In HCOS, CIE supports heterogeneous infrastructure environment, including common physical infrastructure (computing, networking and storage devices), virtualized infrastructure (virtualized computing, networking and storage devices) and other cloud infrastructure platform. HCOS can support a variety of mainstream virtualization systems such as the Kernel-based Virtual Machine (KVM), Microsoft Hyper-V and VMware ESXi. At the same time, it also supports physical and virtual resources provided by mainstream public cloud platforms such as Amazon Web Services (AWS), or private cloud platforms such as OpenStack. Furthermore, HCOS implements a comprehensive support for hybrid IT architecture by combining internal and external resource services, based on a combination of internal and public clouds.

2) CI2

As mentioned above, hybrid IT structure is supported in HCOS, so CI2 integrates a variety of different control interfaces, including the driver interfaces of physical resources (host, network, and storage), virtual resources (from different virtualization systems) and the resources of multiple cloud platforms. In this layer, all driver interfaces are described as a general resource object format defined by the ARML, so as to achieve the purpose of uniform resource management.

3) ARML

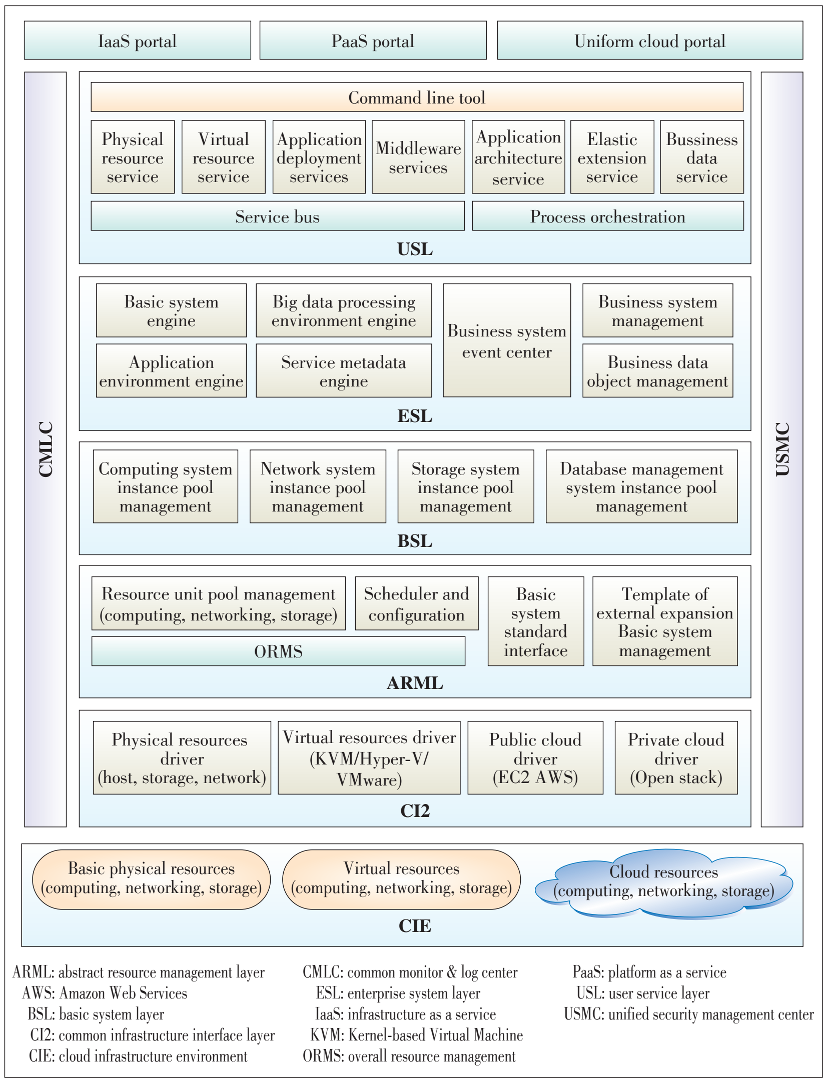
In HCOS, ARML is used to manage and dispatch all resources. The basic facilities in ARML include compute, network and storage resources. The computing unit, the network unit, and the storage unit are defined in this layer. The core object of a computing unit includes the CPU, memory, disk, network card and other components of a host. The core object of a network unit includes switch, router, and IP, and additional components such as load balancing, firewall, etc. The core object of a memory cell includes disk volumes, file systems and block devices. In addition to the management of hardware infrastructure resources, the management of basic software resources is very important.

4) BSL

BSL is the core of HCOS, also called the logic layer. It is mainly used to provide resource instances based on all software resources and collaborative infrastructure resources. All layers under BSL just manage the basic resources, which are not directly available to the cloud application. So all the software and hardware resources are combined in this layer to become the system level resource services to support the operation of cloud applications. Main objects managed in this layer include computing system, network system, storage system and database management system.

5) ESL

In HCOS, ESL works as the kernel layer. It is


IaaS portal
PaaS portal
Uniform cloud portal
CMLC
USMC
Command line tool
Physical resource service
Virtual resource service
Application deployment services
Middleware services
Application architecture service
Elastic extension service
Bussiness data service
Service bus
Process orchestration
USL
Basic system engine
Big data processing environment engine
Business system event center
Business system management
Application environment engine
Service metadata engine
Business data object management
ESL
Computing system instance pool management
Network system instance pool management
Storage system instance pool management
Database management system instance pool management
BSL
Resource unit pool management (computing, networking, storage)
Scheduler and configuration
Basic system standard interface
Template of external expansion Basic system management
ORMS
ARML
Physical resources driver (host, storage, network)
Virtual resources driver (KVM/Hyper-V/VMware)
Public cloud driver (EC2 AWS)
Private cloud driver (Open stack)
CI2
Basic physical resources (computing, networking, storage)
Virtual resources (computing, networking, storage)
Cloud resources (computing, networking, storage)
CIE
ARML: abstract resource management layer
AWS: Amazon Web Services
BSL: basic system layer
CI2: common infrastructure interface layer
CIE: cloud infrastructure environment
CMLC: common monitor & log center
ESL: enterprise system layer
IaaS: infrastructure as a service
KVM: Kernel-based Virtual Machine
ORMS: overall resource management
PaaS: platform as a service
USL: user service layer
USMC: unified security management center

▲**Figure 2. Detailed system architecture of HCOS.**

used to implement practical supports for HCOS applications. The way is to build an application system which can run business applications, and to coordinate the resource allocation for applications in order to achieve the best resource utilization. ESL provides standard application program interfaces (APIs), so that USL can call all functions of this layer.

The functions of ESL can be divided into three kinds: the application deployment, the operation and management of cloud applications, and the management of data. All messages from the three types of functions are processed in a business system event center. The event center is the message bus of HCOS.

6) USL

In HCOS, USL can invoke three kinds of services directly. The first kind is basic system services, such as physical hosts, virtual hosts, virtual networks, object storage, and elastic load balancing service. The second kind of services are on top of the basic system, including programming environment and running environment, such as database service, middleware, application deployment kit, and elastically stretchable application architecture. The third kind is big data processing services and business data services. This is especially for big data applications.

USL has a process orchestration module, which can organize the order of multiple services to realize user specified workflow. All services are registered on the HCOS service bus. The service bus receives these registrations from the APIs invocated in the corresponding service layer. Based on the abilities of the service bus and process orchestration, various services can be combined to form complete business applications. This approach provides a flexible extension of human-computer interaction. Users can redefine the business model of COS according to their actual demands.

In addition, USL includes a command-line tool which is similar to the shell command in a traditional OS. It is also a basic and important human-computer interaction tool, which provides a means of interacting with HCOS where the user (or client) issues commands to the system in the form of successive lines of text (command lines). Users can call all functional APIs of HCOS kernel layer through entering Python commands into the command-line tool. Moreover, HCOS allows users to call functional APIs with other programming languages by building other forms of command-line tools or graphical portals.

7) Common Monitor & Log Center (CMLC)

The functions of CMLC include:

- Log management: providing the monitoring and log management of all components in HCOS
- Physical environment monitoring: providing the monitoring and log management of physical environment
- Virtual environment monitoring: providing the monitoring and log management of virtual environment
- Basic system monitoring: providing the monitoring and log management of various basic systems
- Business system monitoring: providing the monitoring and log management of all instances of business systems
- Service monitoring: providing the monitoring and log management of running and calling of all services.

8) Unified Security Management Center (USMC)

The USMC is the function enhancement of HCOS from security perspective, mainly including user tenant management, role rights management, identity management, red and white list management, security equipment extension, virtualization, security and logs audit, etc. The red and white list management aims at making some constraints on resource access for certain tenants and users. The security equipment extension function supports effective integration of traditional security devices and systems into the basic system layer, such as firewalls, virus protection, intrusion detection, encryption systems and other external systems. Therefore, it can provide comprehensive security protection for business system. Virtualization security includes new security methods achieved with virtualization technology, and new security protections against virtualization security risks. It mainly refers to the security enhancement method implemented in the virtualization layer. The unified security management mechanism integrates a set of security policies adapted to user scenarios to meet different requirements.

2.3 Key Features

Two key features of HCOS are summarized as follows.

1) Unified messaging mechanism

Current IaaS and PaaS platforms are isolated in resource management, leading to a weak performance caused by different messaging mechanisms. We use a new unified messaging mechanism and an adaptive message synchronization to solve those problems. In this way, HCOS realizes efficient and fast synchronization of all resource status.

Specifically, in HCOS, BSL manages all resources and ESL is responsible for business system management. The collaboration between these two components is implemented through the unified messaging mechanism. All the operations from users within HCOS are sent to ESL by USL. Based on this, the resources used by various system instances are notified to the unified message bus when they are instantiated and managed by ARML. At the same time, the unified controller on BSL can be used to collect the information of each system from the message bus through the unified communication mechanism, and complete the control of system. Finally, using the message broker within each system, information is continuously collected to insure the state consistency of the security and reliability of all instances.

Compared to the structure of existing IaaS and PaaS, the whole system of HCOS is coupled by a unified message mechanism. ARML, BSL and ESL work together to achieve the unified management of resources, business system and basic system instances. As soon as a system instance states changes, each layer’s components can automatically adjust to ensure the consistency of business system. The new architecture and message mechanism of HCOS eliminate the boundary of traditional IaaS and PaaS systems, solving the problem that current IaaS and PaaS systems are not consistent in state management.

2) Unified scheduling mechanism

The computing, storage and other resources of current cloud platforms are scheduled independently, which affects the overall scheduling based on the characteristics of applications and leads to low efficiency of resource usage. Moreover, the global optimization cannot be achieved due to the separation of resource adjustment and application demand.

HCOS provides a set of system calls to applications in the business system layer, based on which the application system can define and use resources in the cloud environment flexibly. For the process of application construction, first of all, BSL specifies the resource requirements and the dynamic scheduling strategy of application system in a formal description way. An automated application system construction process is then realized combined with the business characteristics and scheduling strategy. The running process of application system adjusts the resource allocation of BSL according to the operating conditions of application in ESL, so as to achieve the purpose of dynamic optimization of application system resources. For heterogeneous resources, our system has an adapted scheduling strategy.

In addition to run-time scheduling, HCOS also supports presetting a series of behavior patterns or learning new behavior patterns for business systems in running time, and then schedules the resource according to the behavior patterns. The management of behavior patterns reflects the ability of our intelligent mode management system. Usually there are more than one business systems in HCOS, so the scheduling decision must rely on the state of all business systems. If there is no resource competition for two business systems, the policy of as little as possible should be used to ensure the normal operation of these two business systems. Resource requirements of multiple business systems should be considered while scheduling are mainly reflected in the intelligent behavior management system. If users can set the default behavior pattern of the business system, at the time of system deployment, several engines can be triggered when the schedulers are dispatching tasks.

The innovation of unified scheduling mechanism is realized with multi-objective optimization theory, and the intelligent perception approach to user policy, business characteristics and heterogeneous resources features. It breaks through the traditional independent scheduling of resources and services.

3 Performance Evaluation

3.1 Performance Comparison

Table 1 compares HCOS with VMware's vCloud [4] and Cloud Foundry [5], and OpenStack [6], regarding the fusion of IaaS and PaaS, the heterogeneousness (heterogeneous cloud platforms and resources management) and the unified scheduling mechanism. These three aspects of competence are incomplete with VMware. OpenStack is an IaaS level solution, focusing on platform virtualization and cloud resources management, including virtualization engine, cloud management components, virtual functions and resource pool management, and cloud service components. As a result, it has only some heterogeneous resource management capabilities. Comparing with them, HCOS focuses on the integration of IaaS and PaaS in the effective management of massive hybrid and heterogeneous resources.

▼**Table 1. Comparison of a few COSs**

Features	VMware	OpenStack	HCOS
Fusion of IaaS & PaaS	No	No	Yes
Heterogeneousness	No	Partial	Yes
Unified Scheduling Mechanism	No	No	Yes

HCOS:Hua-Cloud Computing System IaaS: Infrastructure as a Service PaaS: Platform as a Service

3.2 Typical Applications of HCOS

3.2.1 Public Security Cloud Based on HCOS

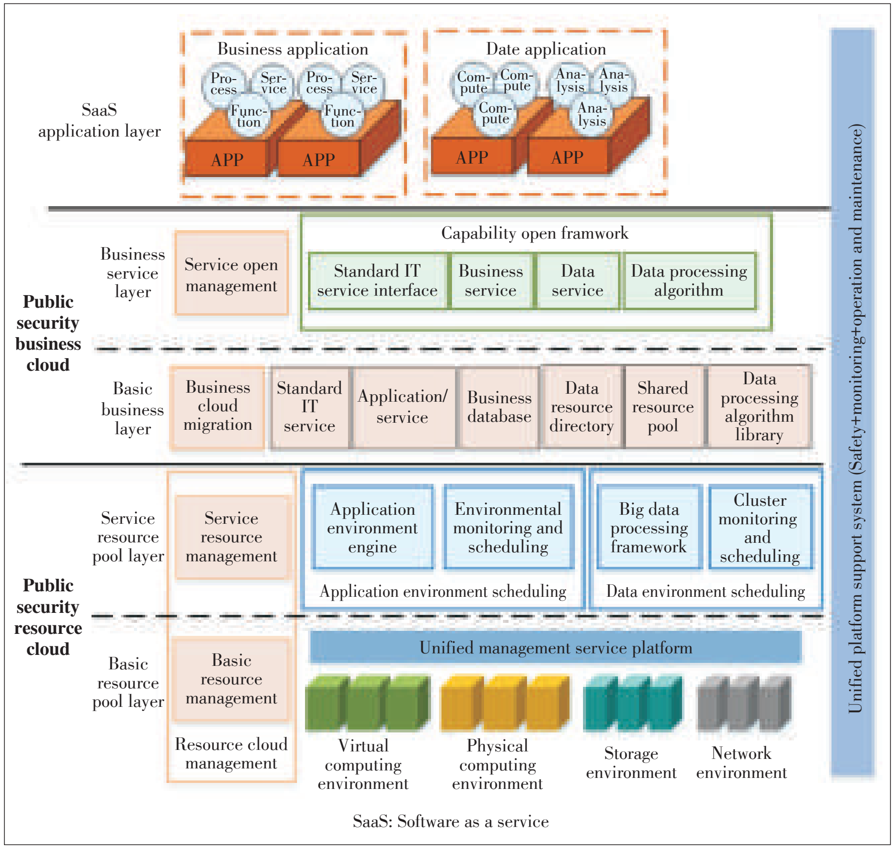
Public Security Cloud based on HCOS is a provincial public security information system, which is consisted in multiple data centers. Public Security Cloud has an overall structure as shown in **Fig. 3**.

The whole Public Security Cloud is divided into three levels. The first level is the public security resource cloud platform, which can provide basic resources, service support, big data processing and other types of services. The second level is the public security business cloud platform, which mainly includes application transforming and data processing services. These two parts are fully supported by HCOS. The kernel and fundamental technology of the second level is integrated with big data management. The third level is a dedicated SaaS layer focusing on public security field, constructed by new generation of public security applications. This Public Security Cloud has been put into use since May 2014.

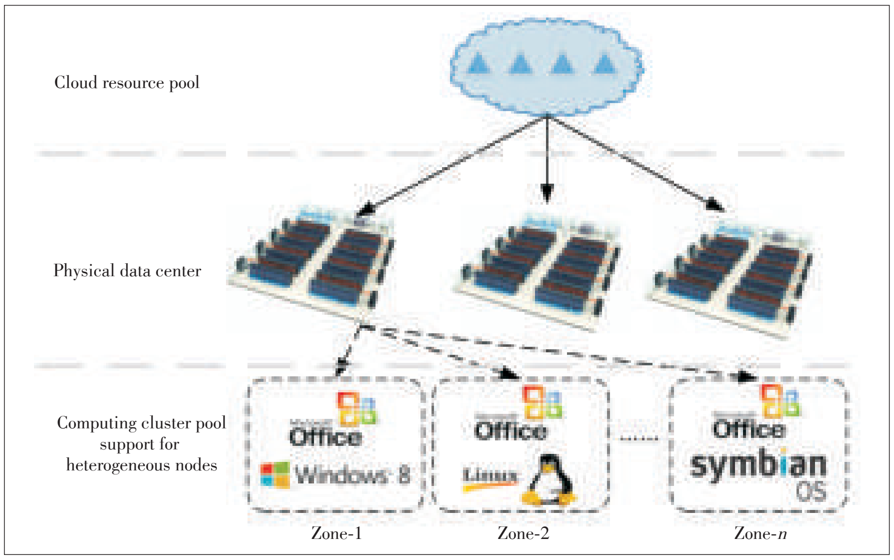
3.2.2 China Unicom Cloud Based on HCOS

As shown in **Fig. 4**, China Unicom Cloud system is a large scale and integrated business cloud platform. It supports heterogeneous nodes to constitute resource pool clusters, providing intelligent resource scheduling, dynamic adaptation, and comprehensive and accurate reforming capabilities for large-scale resource management.

Using a set of unified management interfaces and interactive portals, China Unicom Cloud can realize the management of cloud host and cloud storage resources for design, creation, test, verification, maintenance and recovery in all the life cy-


SaaS application layer
Business application
Pro-cess
Ser-vice
Func-tion
APP
Date application
Com-pute
Ana-lysis
Public security business cloud
Business service layer
Service open management
Capability open framwork
Standard IT service interface
Business service
Data service
Data processing algorithm
Basic business layer
Business cloud migration
Standard IT service
Application/ service
Business database
Data resource directory
Shared resource pool
Data processing algorithm library
Public security resource cloud
Service resource pool layer
Service resource management
Application environment engine
Environmental monitoring and scheduling
Big data processing framework
Cluster monitoring and scheduling
Application environment scheduling
Data environment scheduling
Basic resource pool layer
Basic resource management
Unified management service platform
Resource cloud management
Virtual computing environment
Physical computing environment
Storage environment
Network environment
Unified platform support system (Safety+monitoring+operation and maintenance)
SaaS: Software as a service

▲**Figure 3. Overall structure of Public Security Cloud by HCOS.**



▲ **Figure 4. General structure of China Unicom Cloud Based on HCOS.**

cle. Snapshot and template methods are adopted, enabling rapid and efficient management.

4 Related Work

In recent years, the research of cloud computing has made big progress. Google has issued GFS [7], BigTable [8] and MapReduce [9]. Amazon has provided EC2 [10] and S3 [11]. Microsoft has raised Windows Azure [12].Some other companies and research institutions also have had their own cloud computing technologies and products. However, the concept and the ability requirement of the cloud computing system were not established at the beginning. Currently, some research focuses on the cloud storage and cloud virtualization. With the emerging technologies mainly concentrating in the performance of the cloud, there were also a few concerns about cloud operating systems [12], [13]. Table 1 provides a comparative study on a few typical COSs. The problems such as the fusion of IaaS and PaaS, unified management for heterogeneous resources, and unified scheduling mechanism have not been well studied.

Amazon [6] aims at providing services at the IaaS layer. Gagana et al. [2] discuss the function provided by the existing PaaS platform with an emphasis on the efficiency of application development. The advantages of the PaaS platform are analyzed by Lawton et al. [14], especially on how PaaS increases the production efficiency of enterprises. Serrano et al. [15] analyze the current state of cloud computing to provide users with help on how to select a suitable infrastructure. Steve et al. [16] present the OpenStack platform to support the heterogeneous framework.

There are quite a few cloud operating systems such as OpenStack [6], VMware vCloud [4], and Cloud Foundry [5]. OpenStack just provides IaaS ability. It can realize effective management of virtualized resources, but does not provide application and service middleware management. VMware vCloud and Cloud Foundry are two independent COS products. The former focuses on the virtualization of hardware resource management, while the latter is a typical PaaS platform. These COSs are closely related to our work and we considered their advantages for the design of our new architecture.

5 Conclusions

This paper presents a unified six-layered cloud operating system referring to the operating system theory of traditional PC. HCOS is designed and implemented based on this new

COS architecture model. In view of the abilities and shortcomings of a few existing COSs, HCOS focuses on the integration of IaaS and PaaS, the heterogeneous platform management and the unified scheduling mechanism. It achieves resource state consistency and fast response based on unified messaging and scheduling mechanism. Users can run any service needed on their devices without concerning themselves with the underlying technologies, and can obtain timely response. HCOS provides a multi-level management structure together with a multipoint structure for heterogeneous resources.

Our future research will focus on the following issues:

1) Improving interface standards of our unified messaging mechanism and strengthening the timeliness of resource scheduling. This work can further help the fusion of IaaS and PaaS. We are also developing parallel scheduling for HCOS to improve the efficiency of resource allocation.
2) Improving the efficiency and intelligent capabilities of big data processing. Current HCOS provides a framework with the unified business system data object that may be distributed in different business applications. We are considering solutions to further improving the efficiency and intelligent processing capabilities of big data with data communication security across different areas.
3) Extending HCOS to more application areas, such as digital home applications. Currently we are evaluating HCOS in the public security cloud and heterogeneous resource cloud with PCs and Tablet computers.

Acknowledgment

The authors would like to acknowledge the helpful comments of anonymous reviews, guest editors and the participants of HCOS projects.

References

[1] A. Wiem and Z. Choukair, "PaaS dependability integration architecture based on cloud brokering," in *Proc. Annual ACM Symposium on Applied Computing*, New York, USA, Apr. 2016, pp. 484–487. doi:10.1145/2851613.2851874.
[2] P. Gagana and V. Sundaresh, "Functional analysis of platform as a service (PaaS)," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 4, pp. 600–603, Apr. 2015. doi:10.17148/IJARCCE.2015.44139.
[3] P. Rana , P. K. Gupta, and R. Siddavatam, "Combined and improved framework of infrastructure as a service and platform as a service in cloud computing," in *Proc.International Conference on Soft Computing for Problem Solving (SocProS 2012)*, India, Dec. 2012, pp. 831–839.
[4] VMware, *VMware vCloud Architecture Toolkit (vCAT): Technical and Operational Guidance for Cloud Success*. California, USA: VMware Press, 2013.
[5] Cloud Foundry. (2017, Mar. 18). *New ecosystem marketplace for cloud foundry*[online]. Available: https://www.cloudfoundry.org
[6] K. Jackson, C. Bunch, and E. Sigler, *OpenStack cloud computing cookbook*. Birmingham, UK: Packt Publishing Ltd, 2015.
[7] S. Ghemawat, H. Gobioff, and S. T. Leung, "The Google file system," *ACM SIGOPS Operating Systems Review*, vol.37, no.5, pp.29–43, Dec. 2003. doi:10.1145/1165389.945450.
[8] F. Chang, J. Dean, S. Ghemawat, et al., "Bigtable: A distributed storage system for structured data," *ACM Transactions on Computer Systems (TOCS)*, vol. 26, no. 2, pp.1–26, Jun. 2008. doi: 10.1145/1365815.1365816.
[9] J. Dean, S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp.107–113, Jan. 2008. doi: 10.1145/1327452.1327492.
[10] Amazon. (2015, Jul. 18). *Elastic compute cloud (EC2) cloud server & hosting*[online]. Available: https://aws.amazon.com/cn/ec2
[11] Amazon. (2015, Jul. 18). *Amazon simple storage service (Amazon S3)*[online]. Available: http://aws.amazon.com/s3
[12] B. Calder, J. Wang, A. Ogus, et al., "Windows azure storage: a highly available cloud storage service with strong consistency," in *Proc. ACM Symposium on Operating Systems Principles*, New York, USA, Oct. 2011, pp. 143–157. doi: 10.1145/2043556.2043571.
[13] F. Zhang, J. Chen, H. Chen, and B. Zang, "CloudVisor: retrofitting protection of virtual machines in multi-tenant cloud with nested virtualization," in *Proc.ACM Symposium on Operating Systems Principles*, New York, USA, Oct. 2011, pp. 203–216. doi: 10.1145/2043556.2043576.
[14] G. Lawton, "Developing software online with platform-as-a-service technology," *Computer*, vol. 41, no.6, pp. 13–15, Jun. 2008. doi: 10.1109/MC.2008.185.
[15] N. Serrano, G. Gallardo, and J. Hernantes, "Infrastructure as a service and cloud technologies," *IEEE Software*, vol.32, no.2, pp.30–36, Mar. 2015. doi: 10.1109/MS.2015.43.
[16] S. Cargo, K. Dunn, P. Eads, et al., "Heterogeneous cloud computing," in *IEEE International Conference on Cluster Computing*, Austin, USA, Sept. 2011, pp. 378–385. doi: 10.1109/CLUSTER.2011.49.

Manuscript received: 2017-05-10

Biographies

CHEN Aiguo (agchen@uestc.edu.cn) received his Ph.D. degree in signal and information processing from Beijing University of Posts and Telecommunications, China in 2009. He was a visiting scholar in Arizona State University, USA from Jan. 2013 to Jan. 2014. He is currently an associate professor at the School of Computer Science and Engineering, University of Electronic Science and Technology of China, China. His main research interests are cloud computing, cyber-physical system, and computer network.

WU Huaigu (fs_whg@msn.com) received his B.S. and M.S. degrees in computer science from University of Electronic Science and Technology of China, China in 1997 and 1999 respectively. He received his Ph.D.degree from McGill University, Canada, in 2008. He is currently a post doctor in the School of Computer Science and Engineering, University of Electronic Science and Technology of China. He worked as an architect in the area of cloud computing for about 10 years. His research interests include distributed information system, cloud and big-data computing architecture, micro-service architecture, and mobile software architecture.

TIAN Ling (lingtian@uestc.edu.cn) received her B.S., M.S. and Ph.D. degrees in computer science from University of Electronic Science and Technology of China, China in 2003, 2006 and 2010, respectively. She is currently an associate professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China. She was a visiting scholar in Georgia State University, USA in 2013. Her research interests include digital multimedia, cloud computing, and big data.

LUO Guangchun (gcluo@uestc.edu.cn) received his B.S., M.S. and Ph.D. degrees in computer science from University of Electronic Science and Technology of China, China in 1995, 1999 and 2004, respectively. He is currently a professor of computer science at the University of Electronic Science and Technology of China. His research interests include computer networking, cloud computing, and big data.

DOI: 10.3969/j.issn.1673-5188.2017.04.004
http://kns.cnki.net/kcms/detail/34.1294.TN.20171027.1436.002.html, published online October 27, 2017

Dew Computing and Transition of Internet Computing Paradigms

WANG Yingwei[1], Karolj Skala[2], Andy Rindos[3], Marjan Gusev[4], YANG Shuhui[5], and PAN Yi[6]
(1. University of Prince Edward Island, Charlottetown, C1A 4P3, Canada;
2. Ruđer Bošković Institute, Zagreb 10000, Croatia;
3. IBM Emerging Technology Institute, Durham 27709, USA;
4. Ss. Cyril and Methodius University in Skopje, Skopje 1000, Macedonia;
5. Purdue University Northwest, Hammond 46323, USA;
6. Georgia State University, Atlanta 30302, USA)

Abstract
The goal of this paper focuses on the development of dew computing, including its origins, research status, development status, and its impact on the transition history of Internet computing paradigms. By gathering and studying all the research papers related to dew computing that we are aware of, we found that these papers can be classified into three groups: dew computing early explorations, dew computing feature research, and dew computing application research. Commercial development in the dew computing area also has progressed fast recently; many dew computing products were developed and put into the market. To distinguish dew computing from other Internet computing paradigms and to reveal its essential features, we analyze the transition history of the Internet computing paradigms from information location and distribution aspects. Online impact and redundancy rate are two indices introduced to perform the analysis. The analysis reveals that dew computing is significantly different from other Internet computing paradigms.

Keywords
dew computing; cloud computing; online impact; redundancy rate; Internet computing paradigm

1 Introduction

Dew computing is a new computing paradigm that appeared after the widely acceptance of cloud computing. Obviously the natural cloud metaphor descends to the ground through the dew metaphor. Since its first appearance in 2015, the concepts of dew and dew computing have been explored by different researchers. Many different definitions and applications have been proposed. Researchers also tried to integrate dew computing into existing applications, such as Internet of Things (IoT) streaming, medical care, indoor navigation, and cyber-physical system. There is a lack of a systematic and comprehensive research on the development history and state-of-art of dew related activities. Additionally, dew computing, as a new Internet computing paradigm, has changed the way PC interact with the network and the cloud; a study regarding to the transition of Internet computing paradigms is needed.

Dew computing is positioned at the ground level in the architecture containing cloud computing, fog computing, and dew computing. The vertical, complementary and hierarchical division from cloud computing to fog computing and to dew computing satisfies the needs of high- and low-end computing demands in everyday life and work. The goal of dew computing is to fully use the potentials of personal computers and cloud services. The dew computing paradigm lowers the cost and improves the performance, particularly for concepts and applications such as the IoT and the Internet of Everything (IoE). In addition, the dew computing paradigm will require new architectural and programming models that will efficiently reduce the complexity and improve the productivity and usability of scalable distributed computing.

The contributions of this paper are: 1) systematically survey all the research activities and development activities in the dew computing area; 2) propose two indices to assist further analysis; 3) analyze the transition of the Internet computing paradigm to reveal the essential features of dew computing.

The rest of the paper is organized as the following: Section 2 covers the current status of dew computing research and dew computing applications. In this section, we try to provide a comprehensive survey of the research papers in the dew computing area; we also describe the development work and products related to dew computing. In Section 3, we analyze the Internet computing paradigms at different stages. We also introduce two indices to assist such analysis. Section 4 is the conclusions.

2 Dew Computing and Its Applications

In this section, we collect and study all the existing important research work in the dew computing area and classify these research works into three categories: early explorations, dew computing features research, and dew computing application research. In this way, we show to the readers the landscape of the dew computing research area. We also describe the progress of commercialized dew computing applications.

2.1 Early Exploration

The first group of research work in the dew computing area can be called early exploration. In this group, a few papers were published and the dew computing research area was formed. Cloud-dew architecture [1] was proposed as a possible solution to the offline data accessibility problem. Later, more work about cloud-dew architecture was performed [2]–[4]. Although we may consider cloud-dew architecture as the starting point of dew computing, we found out that dew computing applications started their existence years before the cloud-dew architecture was proposed. This is quite normal in real life: new concepts appear after new products appear; these new concepts generalize and enhance the features of the new products, promote and facilitate the development of broad range of similar products.

At the beginning, the scope of dew computing only includes web applications [1]. In other words, at that time, the proposed dew computing applications were all web applications. Later, a broader definition [5] was proposed. An important work in this area is that a scalable distributed computing hierarchy including cloud computing, fog computing, and dew computing was proposed [6]. Some other work includes the relationships among cloud computing, fog computing, and dew computing [7], the implementation of a horizontal scalable balancer for dew computing services [8], and a new definition and categorization of dew computing [9].

2.2 Dew Computing Feature Research

After the early exploration stage, research work in the dew computing area can be classified into two groups. The first group, dew computing feature research, is discussed in this section; the second group, dew computing application research, is discussed in Section 2.3.

Dew computing feature research focuses on the overall features of dew computing. So far, these research works include the relationship between dew computing and fog computing [10], the implementation techniques of the cloud-dew architecture [6], [11], the relationship between dew computing and cloud computing [12], the relationship between cloud computing, fog computing, dew computing and possible unified framework [13]–[15], and model explorations [16], [17].

2.3 Dew Computing Application Research

Dew computing is not only a research area, but also an application area. Since dew computing is in its emerging stage, many researchers introduce dew computing into various new application areas. Till now, these areas include service and control technology [13], software delivery model [18], high-performance computing [19], life sciences and medical care [20], [21], legacy software language support [22], IoT streaming devices [23], indoor navigation [24], and relations to Cyber-Physical Systems [15], [25].

2.4 Dew Computing Commercial Applications

Dew computing is not only an emerging research area, but also an emerging application area. Some commercial applications already exist. Here we examine the status of dew computing applications through a few examples.

Flight entertainment systems play important roles during flight. While some airplanes have been equipped with screens for all seats, one airline has achieved good passenger satisfaction without these screens. This airline encourages passengers to download an app in their smart phones, tablets, or laptops; using this app, passengers can access free Wi-Fi during the flight. This Wi-Fi will connect passengers' devices to the system on board, and access flight information, movies, TV shows, and so on for free. This arrangement is also called Bring Your Own Device (BYOD) media streaming. Passengers can also access the Internet to surf websites and exchange information with outside world, but fees are normally involved. This on-board entertainment system is a typical dew computing system. It has an on-premises (on-board) dew server to provide services to passengers' devices. This dew server synchronizes with cloud-server (airport server) to get new movies and other information. Another kind of synchronization happens when passengers want to use paid Internet services and it will be accomplished through satellite links.

A few companies are providing series of products for bus Wi-Fi. It seems that these products are also dew computing products. For example, a company in UK, Mobile Onboard Limited, provides this kind of products.

Some music subscription services, such as Spotify, allow users to download music into local devices in a special format and play offline. Such services can also be considered as dew computing services.

The overall concept of dew computing refers to enabling content when there is no Internet connectivity. In early development stages this was initiated by introduction of small wearable devices (dew devices) such as iPod. Some services can be realized as simple downloads and having files on local repository, such as downloading a song, album or film by iTunes, Google play, Spotify, etc. This concept is extended to dynamically created content including maps or web sites. The idea is to enable offline web sites or web pages by providing functionality of a web application provider besides the content availability. For example, this means that the dew device needs to act as a web server, besides downloading static files, as realized by some applications for iOS and Android systems. Google has also implemented this concept on offline maps by realizing a map engine as application provider on the device. The other way of communicating the information is also implemented. Examples include features of uploading content on the web or postponed cloud service requests. In this case, the application will make a temporary copy of the cloud request that will be transferred when the Internet will become available. Shazam is such a service that depends on the Internet, and its later versions include temporal storage of a music sample until it is sent for analysis by an Internet service when the Internet is available. Some existing software packages have dew computing features. For example, open source package Gobby is a collaborative editor. Users can perform editing without the Internet, and the changes made by different users can be synchronized when an Internet connection is available.

It is hard to determine if there is any link between these dew computing applications and the dew computing concept and methodology. One thing is certain: dew computing applications are what users need and have huge potentials. Dew computing research could lead to useful concepts, techniques, frameworks, and platforms; these results will in turn facilitate the further development of dew computing applications.

2.5 Dew Computing Implementation

There are a few efforts on the design and implementation of a dew or the cloud-dew system architecture. Among them, the work in [11] explores the implementation of a dew structure on user's PC and proposes a cloud-dew architecture that features the interaction and collaboration of the cloud and the dew.

Fig. 1 illustrates a conceptual dew structure and the interaction among its components. Note that in some specific applications, more components may be needed. The entire dew structure on the local PC is formed into a dew virtual machine (DVM), an isolated environment for the dew server to be executed on the PC. DVM contains the following components.

- Dew server (DS). The DS is the representative of web servers in the cloud on the local PC. It interacts and periodically synchronizes with web servers and provides customized virtual web service to the user of the local PC.
- Dew analytics server (DAS). Dew analytics server is a local version of the web analytics server, which analyzes data generated when user uses the dew server, viewing dew sites. In this case, user data can be pre-processed before sent to the cloud to the web analytics server.
- Artificial intelligence of the dew (AID). The AID component collects analytical results from the DAS, and uses it to guide the operation of the dew server, customizing the dew server to better serve the local user.

A multi-DVM dew architecture is also proposed in [11], where for each web server the dew interacts, there is a DVM initiated. This design may potentially improve the performance decline caused by the one to fit all scheme. For example, Facebook server and YouTube server may use different database systems. Different settings in the two DVMs to work with them will increase the system performance.

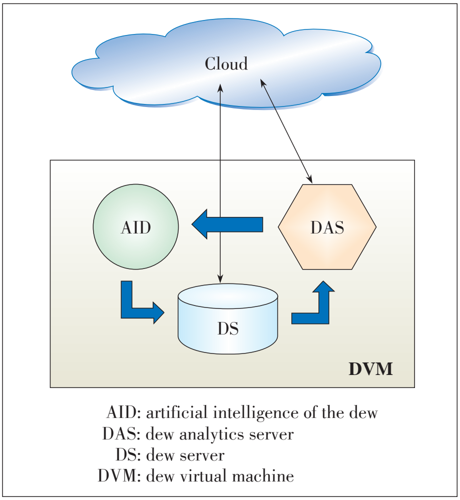

Cloud
AID
DAS
DS
DVM
AID: artificial intelligence of the dew
DAS: dew analytics server
DS: dew server
DVM: dew virtual machine

Figure 1.
Dew components.

3 Transition of Internet Computing Paradigms

In this section, we try to reveal the essence of dew computing through analyzing the transition of Internet computing paradigms.

In terms of general computing paradigms, we may list batch paradigm, time-sharing paradigm, desktop paradigm, network paradigm, and so on. If we concentrate on the Internet, the network paradigm can be further classified from different perspectives. In this paper, we focus on information and the way that information is distributed, propagated, collected, and interacted among the end users through the Internet. Let us briefly review the history of the Internet computing paradigms.

- When ARPANET, the predecessor of the Internet, started in 1960's, there were only 4 nodes. All of them were mainframe computers. At that time, information was saved in each node and the connections between nodes were email and other simple communications.
- With time goes by, personal computers became dominant on the Internet, client-server architecture was proposed, and websites were created. Moreover, cloud computing has pushed the client-server architecture to the extreme: software, platforms, and infrastructures are all placed on the server side; client side local computers are merely terminals to access the Internet. The Internet's architecture has changed significantly since the mainframe computer stage.

- When more and more dew computing applications are developed, the Internet computing paradigm is changed again. Terminal devices, such as laptops, desktops, smart phones, and tablets, are not only terminals any more; they can also provide significant services to the users.

To summarize the above brief review, the early form of the Internet computing paradigm was that information existed in mainframe computers and these mainframe computers communicated with each other through email, File Transfer Protocol (FTP), and other protocols. With the development of the Internet, its architecture changed to websites and cloud computing, communicating via Hypertext Transfer Protocol (HTTP) and FTP like protocols exchanging Extensible Markup Language (XML) structured files. Since dew computing emerged, the Internet's computing paradigm has entered a new stage, where the computers work as conventional web services, or as dew services and directly communicate via XML oriented services and protocols. To simplify our analysis, we only consider these three stages in the development of the Internet computing paradigm: the mainframe stage, the website/cloud computing stage, and the web service/dew computing stage, as presented in **Fig. 2**. Three generations are classified by the information storage locations and level of information exchange.

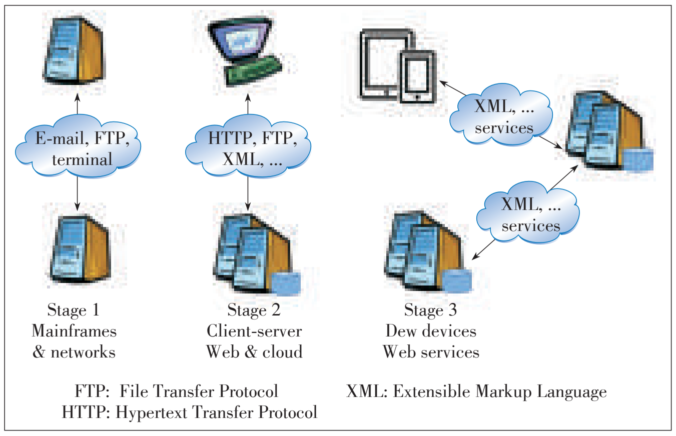
To explore the features of the Internet and the trend of its computing paradigms, we create two indices: online impact and redundancy rate. We will use these indices to describe the Internet computing paradigms and to find the regularities in the transitions of these paradigms.

3.1 Online Impact

In this paper, we consider online impact as an index that measures the importance or impact of the Internet to a node on the Internet. It is defined as

$$I = \log_2 \frac{U}{V}, \tag{1}$$

where I is the online impact of a node on the Internet; U indicates the amount of information available to a node if this node is online; V is similar to U except that when the node is offline.



▲**Figure 2. Development stages of Internet computing paradigms analyzed by information distribution.**

In the above definition, U and V are information amounts, but we did not specify how to determine the values of U and V. The basic way to determine an information amount is to find the size of the related storage in bytes. For example, if a cell phone has 16GB storage and the storage is almost full, we may estimate that $V = 2^{34}$ bytes.

Now, we use an example to show how to calculate online impact I. The first question we have to solve is how to calculate U, that is, how to determine the total amount of available information when a device is online. Note that we are not talking about the total amount of information that has ever produced and saved on the Internet; we are talking about the amount of information that is publically available and relevant. Also, the accuracy of this number does not matter much because it is only used in an example. For these reasons, we chose a simple number as the base of our estimation: the sum total of data held by Google, Amazon, Microsoft, and Facebook is at least 1200 petabytes [26]. For easy estimation, we assume that the sum total of all available data held on the Internet is 8192 petabytes, i.e., $U = 2^{63}$ bytes. Next we need to estimate V. We assume that the node in question is a personal computer and it has 1TB storage; the storage is almost full. Under these assumptions, $V = 2^{40}$ bytes. Thus the online impact

$$I = \log_2 \frac{2^{63}}{2^{40}} = 23. \tag{2}$$

In some special situations, we may also calculate the online impact using other information estimation methods, not the basic one discussed above. For example, a user is interested in a special group of programs, say games. Suppose there are 1000 such programs on the Internet, and there are 8 such programs on the local node. We may use the number of available programs to estimate the available information amounts: $U = 1000$ and $V = 8$. This node's online impact would be

$$I = \log_2 \frac{1000}{8} = 7. \tag{3}$$

For the same node on the Internet, we may get different online impact numbers using different information amount estimation methods. The reason we define the online impact in such an indefinite way is because it is a subjective index: for the same node, when the Internet becomes not available, it could have a great impact to one user but have a less severe impact to another user.

Although the online impact is defined in an indefinite way, it does not prevent us from using this index to perform comparisons and other analysis because online impact numbers are comparable as long as they are all calculated using the same method. The freedom of choosing information amount estimation method makes the online impact meaningful in different situations. Once an information estimation method is chosen, this method should be applied consistently so that this index would be comparable.

Now we try to estimate the online impact values for different stages of Internet development: the mainframe stage, the cloud computing stage, and the dew computing stage. In the following analysis, basic estimation method is used.

In the mainframe stage, major computing tasks were done by each node. The Internet (ARPANET) provides email, telnet, and similar services. Since these services were low-efficient, they cannot drastically change the size of the available information. We can estimate that the amount of the available information when a node was online could be more than the amount of the available information when this node was offline, but would not be more than doubled. Thus, the online impact would not be more than 1. Generally speaking, in the mainframe stage, the online impact was pretty low.

In the cloud computing stage, major computing tasks are done at the server side. In extreme situations, the node on the user side (user's device) is merely a terminal; no useful information is stored in the node. In these situations, the user expects the device is always online; when the device is offline, the user's device loses its connection and control to the cloud; although submitted jobs can still be processed, no more new jobs can be submitted. When we calculate online impact for the extreme situations, we will find that U is a big number and V is 0 or almost 0, which leads to an online impact that is infinity or a huge number.

In the dew computing stage, the dew server and related databases on a node provide useful information to the user. Apparently, the whole dew mechanism increased the V value, and thus decreased the online impact: either from infinity to a finite number or from a bigger number to a smaller number.

Next, we summarize the above three situations:

In the early mainframe stage, the online impact was very low; it shows that the Internet had not well developed and the Internet did not have vital importance at that time.

Currently, cloud computing is still the dominant form of the Internet computing paradigm. In other words, the Internet is still in the cloud computing stage. In this stage, a node's online impact is very high and, in theory, could be infinity. This fact reflects that the Internet has well developed; the Internet has vital importance to many users; once offline, some users feel they can do nothing.

In dew computing stage, the online impact is getting smaller. This is an interesting phenomenon. In the past, the online impact is getting bigger and bigger, which shows that the Internet is getting more and more important. Now the trend has changed: the online impact is getting smaller; it seems that the Internet is getting less important.

Does dew computing reduce the importance of the Internet? If we carefully analyze this issue, we will find that the answer to this question is no. Dew computing does reduce the online impact number; in other words, dew computing reduces the shock when an Internet connection is lost. However, this benefit is obtained based on the support of the Internet. We may notice that the dew computing components on a user device have collaborative relationship with cloud servers. In the past, a node's available information may have nothing to do with the Internet. If dew computing paradigm is adopted, a node's available information, i.e. the dew component, would be well organized and supported by the Internet. In this way, we may say that the Internet plays a more important role in the dew computing stage.

This brings in an important conclusion: dew computing reduces the online impact, thus it reduces the user's shock when an Internet connection becomes unavailable. Dew computing does not reduce the importance of the Internet; on the contrary, dew computing makes information on users' devices systematically organized and collaborative with the Internet. Dew computing enhances the Internet and cloud computing.

3.2 Redundancy Rate

The redundancy rate is an index that measures how much information on a node has a copy on the Internet.

Redundancy rate is defined as:

$$R = \frac{W}{V}, \tag{4}$$

where R is the redundancy rate, V is the node's available information amount, and W is the amount of information inside V that has a copy on the Internet. All the discussions in Section 3.1 regarding to the methods to determine values U and V are also suitable here for values W and V.

In the mainframe stage, major computing tasks were done by each node; programs and other form of information were also saved at each node. Redundancy copy may exist due to special arrangements, but there were no generally-available redundancy copies. The redundancy rate at this stage was very low.

In the cloud computing stage, major computing tasks are done at the server side; useful information is also saved on the server side. The node on the user side (user's device) is merely a terminal, no useful information is stored on the node; it is not necessary to make a redundant copy for such information. Even though important information is stored on the node, redundant copy of such information might be specially arranged; there is no systematic mechanism to perform redundant copying. Generally speaking, the redundancy rate at this stage is also very low.

In the dew computing stage, the dew server and related databases on a node provide useful information to the user. These dew components are supported by the cloud servers and synchronized with the cloud servers; these dew components all have redundancy copies on the Internet. If the node has completely adopted dew computing, all the available information in this node would be dew component information. Since all the dew components have redundant copies on the Internet, the redundancy rate would be 100%. If the node also contains information other than that related to dew computing, the redundancy rate could be less than 100%. We can conclude that dew computing significantly increases the redundancy rate. If dew computing components are dominant on the node, the redundancy rate could be close to 100%.

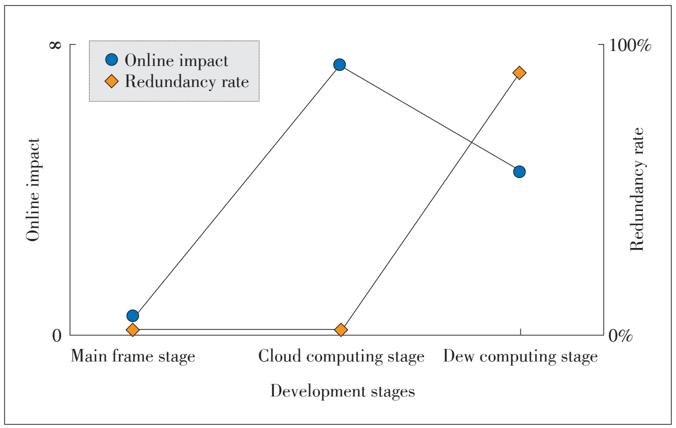
3.3 Analysis of Online Impact and Redundancy Rate

In the above two sections, we introduced two indices: online impact and redundancy rate. These two indices change over the three stages of Internet computing paradigm development: mainframe stage, cloud computing stage, and dew computing stage. We summarized these changes in **Fig. 3**.

We notice that the online impact has increased from almost 0 in the mainframe stage to a big number close to infinity in the cloud computing stage, and then decreased in the dew computing stage. The increasing portion shows that the Internet has developed greatly such that users heavily rely on it. Once the Internet is not available, the impact to users is very big. In some extreme situations, users feel nothing can be done without the Internet. While the high online impact number shows the greatness of the Internet, we should also notice the other side of the story: even if the Internet is just temporarily not available, we have to face some negative effects. The decreasing portion of the online impact line reflects the efforts to solve this problem.

In the dew computing stage, online impact is reduced because when the Internet becomes not available, the dew server can provide some services to the user so that the user will feel a less severe impact.

Let us further apprehend the meaning of the index online impact. Is it correct to say that the bigger the online impact, the better the Internet? This conclusion probably was right for the past few decades. To some extent, online impact is an index to measure the greatness of the Internet. With the development of the Internet, more and more data and services have been added to the Internet; users become more and more reliant to the Internet; the online impact becomes higher and higher.



▲Figure 3. Online impact and redundancy rate change over major stages of Internet computing paradigm development.

In our daily lives, we rely on many resources, such as utilities, transportation, governments, and heroes. When some resources are not available, we will feel some impact on our lives. The bigger the impact is, the more important the resource is. For example, we heavily rely on electricity; we will feel a big impact when a blackout happens. While we all agree that electricity is vital to our lives, we feel unsafe if electricity is the only means we can count on. Homeowners may want to buy generators or propane tanks/stoves in winter as backup. In other words, homeowners want to reduce the impact when a blackout happens.

Generally speaking, people would like to use the resources that we highly rely on because these resources must be of great values, but people also want to have some backup options so that they would not get into a big trouble when these great resources become not available. In other words, people like resources that have great impact factors, but do not want these impact factors to be too big.

In the case of the Internet, users want devices on the Internet to have big online impact numbers, but they do not want these numbers to be too big. Dew computing is a solution to reduce the online impact: from infinity to a finite number or from a bigger number to a smaller number.

Analyzing the redundancy rate line in Fig. 3, we notice that the redundancy rate stays quite low in the mainframe stage and the cloud computing stage, and then increases to a high number close to 100% in the dew computing stage. The shape of this line shows that dew computing has brought in something new. This new feature can be described as user-aware (systematic) redundancy.

In computer science, redundancy is a widely-used technique to improve reliability and availability, but this technique was mainly used by developers and was often transparent to users. Dew computing systematically brings in redundancy technique to users. In a dew computing system, the user is aware that data and services are redundantly installed both in his/her own device and in the cloud servers; the user is also aware that his/her device will exchange data with cloud servers when it is possible and necessary.

Dew computing is not only a technique to implement applications; it also brings in a new feature, user-aware redundancy, to the users and familiarizes users to use this new feature. In this sense, dew computing brings in significant changes to the Internet computing paradigm.

To summarize the whole analysis using these two indices, online impact and redundancy rate, we obtain the following observations:

1) Dew computing is a new form of the Internet computing paradigm, which is significantly different from previous Internet computing paradigm forms.
2) Dew computing enables users to feel a less severe impact when the Internet becomes not available.

3) Dew computing brings in a new feature, user-aware redundancy, to users.

4 Conclusions

Since the first group of papers related to dew computing were published in 2015 and early 2016, dew computing research work can be roughly classified into three major groups: early exploration, dew computing feature research, and dew computing application research. Considering the short time since this research area was started, it is amazing that so many papers have been published already. It is worth to note that among the recently published papers, 7 papers were related to the overall features of dew computing, but 10 of them were related to dew computing applications. This phenomenon indicates that dew computing application is the current focus of dew computing research.

To be consistent with the above dew computing research focus, commercialized dew computing applications have also progressed pretty fast recently. Many dew computing products have been developed and put into the market. This fact further indicates that dew computing is needed by the society and the market.

Dew computing has been going through fast progress in both research and development, but we still need to answer the following questions: what are the essential differences between dew computing and other Internet computing paradigms? What are the special features of dew computing? The Internet computing paradigm transition analysis was performed to answer these questions. Online impact and redundancy rate are two indices introduced to perform such analysis.

The analysis revealed the following two features of dew computing: 1) Dew computing enables users to feel a less severe impact when the Internet becomes not available; 2) dew computing brings in a new feature, user-aware redundancy, to users. These two features indicate that dew computing is significantly different from other Internet computing paradigms.

References

[1] Y. Wang, "Cloud-dew architecture," *International Journal of Cloud Computing*, vol. 4, no. 3, pp.199–210, 2015. doi: 10.1504/IJCC.2015.071717.
[2] Y. Wang and Y. Pan, "Cloud-dew architecture: realizing the potential of distributed database systems in unreliable networks, " in *21st International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA' 15)*, Las Vegas, USA, Jul. 2015, pp. 85–89.
[3] D. Bradley. (2016, May 8). Dew helps ground cloud services [Online]. Available: http://sciencespot.co.uk/dew-helps-ground-cloud-services.html
[4] Z. Kang, "A new method to implement ldns in the cloud-dew architecture, " University of Prince Edward Island, Canada, CSIT Research Rep. CS-21, , Nov. 17, 2005.
[5] Y. Wang. (2015, Nov. 10). The initial definition of dew computing, dew computing research [Online]. Available: http://www.dewcomputing.org/index.php/2015/11/10/the-initial-definition-of-dew-computing
[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] Y. Wang. (2015, Nov. 12). The relationships among cloud computing, fog computing, and dew computing, dew computing research [Online]. Available: http://www.dewcomputing.org/index.php/2015/11/12/the-relationships-among-cloud-computing-fog-computing-and-dew-computing
[8] 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.
[9] Y. Wang, "Definition and categorization of dew computing," *Open Journal of Cloud Computing (OJCC)*, vol. 3, no. 1, pp. 1–7, 2016.
[10] T. Mane. (2016, Jul. 2). Fog-dew architecture for better consistency [Online]. Available: https://eye3i.wordpress.com/2016/07/02/adressing-inconsistency-in-dew-computing-using-fog-computing
[11] D. E. Fisher and S. Yang, "Doing more with the dew: a new approach to cloud-dew architecture," *Open Journal of Cloud Computing (OJCC)*, vol. 3, no. 1, pp. 8–19, 2016.
[12] A. Rindos and Y. Wang, "Dew computing: the complementary piece of cloud computing," in *IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)*, Atlanta, USA 2016, pp. 15–20. doi: 10.1109/BDCloud-SocialCom-SustainCom.2016.14.
[13] Z. Šojat and K. Skala, "Views on the role and importance of dew computing in the service and control technology," presented at Distributed Computing, Visualization and Biomedical Engineering Conference, Part of the 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (DC VIS 2016) , Opatija, Croatia, 2016.
[14] Z. Sojat and K. Skala, "The dawn of dew: dew computing for advanced living environment," in *DEWCOM 2017*, Opatija, Croatia, May 2017, pp. 375–380.
[15] M. Frincu, "Architecting a hybrid cross layer dew-fog-cloud stack for future data-driven cyber-physical systems," in Proc. *MIPRO DEWCOM 2017*, Opatija, Croatia, May 2017, pp. 427–431.
[16] P. Brezany and F. Khan, "Cloud-dew data provenance framework," presented at DEWCOM 2017, Opatija, roatia, May 2017.
[17] Y. Wang, "An attempt to model dew computing," presented at DEWCOM 2017, Opatija, Croatia, May 2017.
[18] Y. Wang and D. LeBlanc, "Integrating SaaS and SaaP with dew computing," in *IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)*, Atlanta, USA, 2016, pp. 590– 594. doi: 10.1109/BDCloud-SocialCom-SustainCom.2016.92.
[19] G. Oparin, V. Bogdanova, S. Gorsky, and A. Pashinin, "Service-oriented application for parallel solving the parametric synthesis feedback problem of controlled dynamic systems," in *Proc. MIPRO DEWCOM 2017*, Opatija, Croatia, May 2017, pp. 381–386.
[20] Y. Gordienko, S. Stirenko, O. Alienin, et al., "Argmented coaching ecosystem for non-obtrusive adaptive personalized elderly care on the basis of cloud-fog-dew computing paradigm," in *Proc. MIPRO DEWCOM 2017*, Opatija, Croatia, May 2017, pp. 387–392.
[21] P. Brezany, T. Ludescher, and T. Feilhauer, "Cloud-dew computing support for automatic data analysis in life sciences," in *Proc. MIPRO DEWCOM 2017*, Opatija, Croatia, May 2017, pp. 392–398.
[22] N. Crnko, "Distributed database system as a base for multilanguage support for legacy software," in *Proc. MIPRO DEWCOM 2017*, Opatija, Croatia, May 2017, pp. 399–402.
[23] m. gusev, "A dew computing solution for IoT streaming devices," in *Proc. MIPRO DEWCOM 2017*, Opatija, Croatia, May 2017, pp. 415–420.
[24] D. Podbojec, B. Herynek, D. Jazbec, et al., "3D-based Location Positioning Using the Dew Computing Approach for Indoor Navigation, Proceedings of MIPRO 2017," in *DEWCOM 2017*, Opatija, Croatia, May 2017, pp. 421–426.
[25] T. Lipic and K. Skala, "The key drivers of emerging socio-technical systems: a perspective of dew computing in cyber-physical systems," presented at *DEWCOM 2017*, Opatija, Croatia, May 2017.
[26] G. Mitchell. (2017, Jun. 10). How much data is on the internet? [Online]. Available: http://www.sciencefocus.com/qa/how-many-terabytes-data-are-internet

Manuscript received: 2017-06-16

Biographies

WANG Yingwei (ywang@upei.ca) received his B.S. and M.S. degrees from Harbin Institute of Technology, China. He received his Ph.D. from the University of Waterloo, Canada. From 1982 to 1997, he worked at Harbin Institute of Technology as a Research Assistant, a Lecturer, and an Associate Professor. In 1999, he worded at the University of Waterloo as a Lecturer. From 2003 to 2004, he worked as a postdoctoral research associate at the University of Western Ontario. Since 2004, he has worked at the University of Prince Edward Island, Canada. Dr. Wang's research interests include dew computing, cloud computing, internet of things, and bioinformatics. He was awarded twice for research achievements by his university. His work in cloud-dew architecture lead to the creation of the dew computing research area.

Karolj Skala (skala@irb.hr) is a senior scientist at the Ruđer Bošković Institute, Croatia and the head of Centre for informatics and Computing there . He has been a lecturer at University of Zagreb, Croatia since 1989. Dr. Skala was Chairman of the International Scientific Symposium on Data and Life Sciences Based on Distributed Computing. He is a member of the MIPRO programme committee, and a member of the COGAIN association. Besides, he is the national coordinator in the European Cooperation in Science and technology 4 COST projects, a project member coordinator of the 5 EU FP6, and the national project leader of 6 EU FP7 and 4 EU Horizon2020 projects. He is also a member of Croatian Academy of Technical Science, and an associate member of Hungary Academy of Science. He was awarded the Annual Science Prize of the Hungarian Academy of Sciences in 2015 and the State Awards for Science of Croatia in 2016.

Andy Rindos (rindos@us.ibm.com) is currently the program director for Industry Verticals (Public Sector, Health, Finance), Strategic Customer Success, Watson and Cloud Platform at IBM Emerging Technology Institute, USA. He also heads the Research Triangle Park Center for Advanced Studies (CAS; IBM North Carolina university relations) and the WW CAS network, as well as US university relations and the IBM Cloud Academy. Most recently, he was the program director for the Emerging Technology Institute for IBM Cloud (previously the IBM Middleware Chief Technology Office), and has previously headed the WebSphere Technology Institute as well as performance for Tivoli and Networking Hardware divisions. He is a member of the IBM Academy of Technology, as well as an NC State Adjunct Associate Professor. He joined IBM in 1988, after receiving his Ph.D. in electrical engineering from the University of Maryland, USA. Prior to IBM, he was a Neurophyisologist at the National Institutes of Health in Bethesda MD, USA.

Marjan Gusev (marjan.gusev@innovation.com.mk) is a professor at Ss. Cyril and Methodius University in Skopje, Macedonia, Faculty of Computer Science. He has participated in, coordinated and evaluated many national IT projects and participated and coordinated more than 30 TEMPUS, FP6, FP7 and H2020 projects of the European Commission. He has published over 500 papers in the area of parallel processing, cloud computing, security, computer networks and high-performance computing. He received the award and certificate for the Best Scientist and Researcher at the Ss Cyril and Methodius in 2012 and the Best Professor Award in 2015. Also he was awarded the IEEE EDUCON Best Paper Award in 2013 and the 8th Annual SEERC DSC2013 Best Paper Award.

YANG Shuhui (shuhuiyang@pnw.edu) received her B.S. degree from Jiangsu University, China, and her M.S. degrees from Nanjing University, China. She received her Ph.D. from Florida Atlantic University, USA. From 2007 to 2009, she worked as postdoctoral research associate at Rensselear Polytechnic Institute. Since 2009, she has worked at Purdue University Northwest. Dr. Yang's research interests are wireless communication, and distributed systems. She has published more than 40 papers in her fields. She has served in organizing committees and technical program committees for many IEEE conferences.

PAN Yi (yipan@gsu.edu) is currently a Regents' Professor and Chair of Computer Science at Georgia State University, USA. He has served as an Associate Dean and Chair of Biology Department during 2013-2017 and Chair of Computer Science during 2006-2013. Dr. Pan joined Georgia State University in 2000, was promoted to full professor in 2004, named a Distinguished University Professor in 2013 and designated a Regents´ Professor (the highest recognition given to a faculty member by the University System of Georgia) in 2015. Dr. Pan received his B.Eng. and M.Eng. degrees in computer engineering from Tsinghua University, China, in 1982 and 1984, respectively, and his Ph.D. degree in computer science from the University of Pittsburgh, USA, in 1991.

DOI: 10.3969/j.issn.1673-5188.2017.04.005
http://kns.cnki.net/kcms/detail/34.1294.TN.20171023.1016.002.html, published online October 23, 2017

Online Shuffling with Task Duplication in Cloud

ZANG Qimeng[1] and GUO Song[2]
(1. School of Computer Science and Engineering, The University of Aizu, Aizu-Wakamatsu 965-0006, Japan;
2. Department of Computing, The Hong Kong Polytechnic University, Hong Kong SAR 852, China)

Abstract

Task duplication has been widely adopted to mitigate the impact of stragglers that run much longer than normal tasks. However, task duplication on data pipelining case would generate excessive traffic over the datacenter networks. In this paper, we study minimizing the traffic cost for data pipelining task replications and design a controller that chooses the data generated by the first finished task and discards data generated later by other replications belonging to the same task. Each task replication communicates with the controller when it finishes a data processing, which causes additional network overhead. Hence, we try to reduce the network overhead and make a trade-off between the delay of data block and the network overhead. Finally, extensive simulation results demonstrate that our proposal can minimize network traffic cost under data pipelining case.

Keywords

cloud computing; big data; shuffling; task duplication; traffic

1 Introduction

Driven by technology and economies of scale, cloud computing platforms are becoming the mainstream hosting platform for a variety of infrastructure services and data intensive applications [1]. With the development of cloud computing, large scale data computing has gained widespread attention due to its ability to automatically parallelize a job into multiple short tasks, and transparently deal with the challenge of executing these tasks in a distributed setting. The crux is that the execution time of a task is variable. The tasks on the slowest machines (stragglers) become the bottleneck in the completion of a job. Straggler mitigation has received considerable attention across prior studies [2], [3]. Launching copies for the slower tasks is the dominant technique to mitigate the stragglers. A fundamental limitation is that traditional techniques for straggler mitigation do not take into account of network traffic in the pipelining case. While this approach of replicating tasks decreases task completion time, it may increase network traffic. The wide area network (WAN) that connects geographically distributed machines is one of the most critical and expensive infrastructures that costs hundreds of millions of dollars annually [4].

Thus, providing traffic awareness for pipelining task duplications is essential for big data processing applications in cloud. To the best of our knowledge, however, no existing work is in place to respect the network traffic in the pipelining case. We seek for a mechanism to minimize the network traffic cost by coordinating task duplication and pipelined intermediate data.

In this paper, we introduce a controller to manage task duplications. Every task duplication has a communication link with the controller. To reduce network traffic, the controller picks the first finished task output from a duplication and discards the subsequent data generated later by other duplications. Each task duplication produces a large number of key-value pairs for pipelining data records. The communication between controller and each duplication will greatly increase the network overhead. To solve this problem, we propose an online distributed algorithm which is executed by each task duplication. The algorithm makes a rule to group data records and transmit the group instead of transmitting one data record at a time. This solution helps reduce the communication times with the controller, but may cause delay. We thus make a trade-off between the transmitting delay and communication overhead.

The remainder of the paper is structured as follows. Section 2 reviews the parallel computing and related work. Section 3 describes the motivation and challenges of the proposed work. Section 4 gives an overview of our system model and formulates the network traffic minimization problem. Section 5 introduces online distributed algorithms. Section 6 demonstrates the performance evaluation results. Finally, section 7 concludes this work.

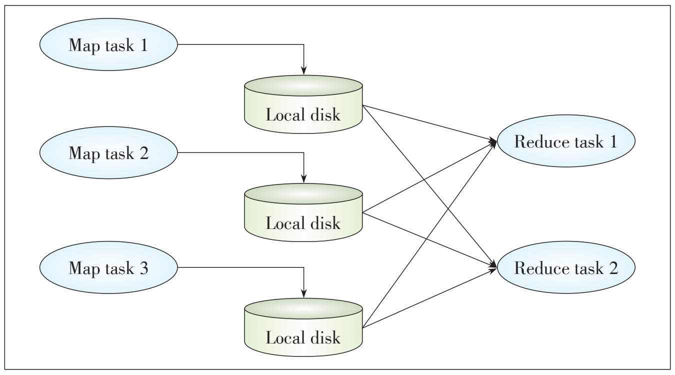
2 Background and Related Work

MapReduce [5] has emerged as the most popular programming framework for big data processing in cloud. Ever since it was introduced by Google, the MapReduce programming model has gained in popularity, thanks to its simple, yet versatile interface. The paradigm has also been implemented by the open source community through the Hadoop project [6], maintained by the Apache Foundation. MapReduce divides a computing job into two main phases, namely map and reduce, which in turn are carried out by several map tasks and reduce tasks. The intermediate data transmission process from map task to reduce task is referred to as shuffling. We use a typical MapReduce job to show how the model works. As shown in **Fig. 1**, several map tasks are executed at the same time. The generated intermediate results in forms of key-value pairs are stored in local storage. To simplify fault tolerance, MapReduce materializes the output of each map before it can be consumed. A reduce task cannot fetch the output of a task until the map has finished executing and committed its final output to disk.

A fundamental limitation of MapReduce is that the intermediate data is materialized. Storing intermediate data on the local disk impacts MapReduce's performance. Thus, optimizing the intermediate data management is a popular topic in industry and academic research. In order to improve shuffling performance, a modified MapReduce architecture is proposed, MapReduce Online, in which intermediate data is pipelined between operators [7]. When a map task generates a key-value pair, it immediately sends this key-value pair to corresponding reduce task, avoiding intermediate data storage. More- over, after receiving key-value pairs, reduce tasks can start data processing earlier, without waiting map tasks to finish. Pipelining intermediate data improves system utilization.

The technique of replicating tasks has been widely applied in parallel computing by system designers [8].

By adopting task duplication, the distributed computing systems achieve predictable performance. Several large scale applications use the technique of task duplication to mitigate stragglers, such as Longest Approximate Time to End (LATE) [9], Dryad in Microsoft [10] and Mantri [11]. When the fastest task finishes, its output will be stored and then other running task copies will be killed. The original MapReduce paper handles the stragglers by adopting backup tasks.



▲**Figure 1. The MapReduce model.**

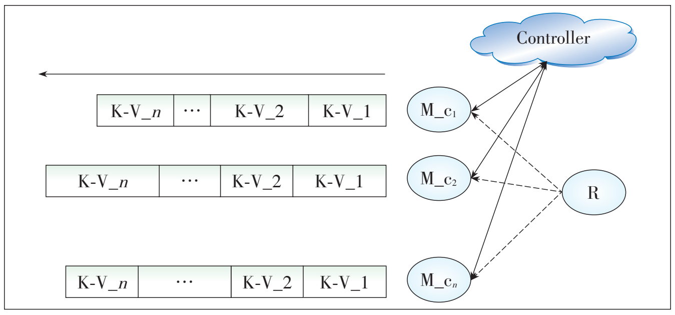
However, the traditional techniques of straggler mitigation causes excessive network traffic when the pipelining case is used for them. Replicating tasks in large scale applications has a long history [12], [13] with extensive studies in prior work. However, these studies are not applicable in the pipelining case. For example, we suppose that multiple task copies are launched for a map task with pipelined intermediate data. Whichever task copy produces intermediate data, the reduce task will receive the data, even though it has received the data from other task copy already, which generates huge data transmission in the shuffling process. The most important difference among our work and the related work is that our proposed method respects the network traffic. Since the limited bandwidth shared by multiple applications, the redundant data transmission would congest the network and bring negative performance for cloud computing applications. To fill in this gap, we take a stab at task duplication for pipelining data transfer.

3 Motivation

Here, we illustrate the value of task duplication in the pipelining case and give a reasonable solution.

When big data processing is executed on cloud, these jobs consist of many parallel tasks. Every task is executed in parallel on different machines. In our proposed work, there is a controller to manage the execution of each task replication. In the same way, every task replication runs in parallel on different machines. The controller will receive a notification when a machine finishes its assigned task, and then choose the first finished task to transmit the data. The data generated by other task replications will be discarded after the controller decides one of the task replications to transmit the data records. By adopting this method, the overlapping data will not be transmitted, which minimizes the network traffic cost generated by same duplications. As shown in **Fig. 2**, multiple task copies are launched for one of the map task, i.e. M_c1, M_c2 and M_*cn*. Each task copy produces the key-value pair in a different time. The controller will pick the fastest one for the input of the reduce task.

However, another crucial factor should be taken into account. The machine executing a task replication will communicate with the controller after the task replication generates a



▲**Figure 2. Non-overlapping data block transmitting.**

set of data records. The same task replications will generate great numbers of data records in big data processing. The communication between the controller and task replications greatly increases the network overhead. Let c_c be the cost caused by the communication.

To reduce the cost generated by huge communication, we make a rule to transmit the group of key-value pairs instead of transmitting one key-value pair. Similarly, the controller chooses the first finished the data block group to transmit and discards the data block group generated by other task replications. The challenge is to determine how many key-value pairs should be grouped together. How to divide the data block into groups will be discussed in section 5.

The delay, another crucial fact needed to take into consideration, arises from transmitting the data block group. The data block is not transmitted immediately when it was been generated. If too many key-value pairs are grouped, the data receiving at the next task may have a large delay that seriously slows down the data processing. Otherwise there is a big communication overhead with the controller, almost same with the scheme without grouping. Similarly, the delay causes cost. Let $\min\left(c_c+\sigma\sum_i c_d^i\right)$, be the cost caused by delay. In this paper, we try to make a good tradeoff between c_d and c_c.

Although the approach of combining task duplication and pipelined intermediate data can significantly accelerate data processing, redundant data are generated and may incur congestion on the network. In this paper, we design a novel online shuffling with a traffic controller to eliminate redundant data.

4 Model and Problem Formulation

The MapReduce programming model consists of two primitives: map phrase and reduce phrase. The former phrase produces a set of intermediate records (key/value pairs), which are provided as input to the reduce phrase. As a result, transmitting the intermediate data through the network generates traffic cost. The cost of delivering a certain amount of data over a network link is evaluated by the product of data size and distance between two machines.

Now we formulate the network traffic minimization problem. All symbols and variables used in this paper are shown in **Table 1**. We first consider the data delivering between two machines. Let c_{xy} denote the traffic cost of transmitting data from machine $x \in M$ to machine $y \in R$, which can be calculated by:

$$c_{xy}=v_x d_{xy},\quad x\in M,\quad y\in R. \tag{1}$$

The receiving delay cost of sending group i is c_d^i. To make a good tradeoff between control overhead and data receiving delay, we attempt to minimize the following function:

$$\min\left(c_c+\sigma\sum_i c_d^i\right), \tag{2}$$

Table 1. Symbols and variables

Notations	Description
M	A set of machines to generate intermediate data
R	A set of machines to receive intermediate data
v_x	Data volume of an intermediate record produced by machine x
c_{xy}	The traffic cost from machine x ∈ M to machine y ∈ R
c_c	The cost caused by communication with the controller
c_t^i	The total cost of transmitting group i
c_d^i	The delay cost of group i
d_{xy}	The distance between two machines x and y

where σ is the competitive ratio indicating the relative weight of delay cost [14]. This problem is challenging because we have no knowledge of generation time of key-value pairs in future. Therefore we design an online grouping algorithm that groups key-value pairs in data buffer. The detailed design will be presented in next section.

5 Design of Online Distributed Algorithm

Our proposed online grouping algorithm makes a good tradeoff between control overhead and data receiving delay. In an online big data processing environment, the data is pipelined between operators. The intermediate data should be transmitted to next phrase immediately. In our proposed work, the duplications of a task have the same output, but the time to produce each intermediate data record is different. By using a controller, the task duplications can communicate with the controller to confirm whether it transmits or discards its output data.

Another issue we need to solve is that the large number of communication times between a task duplication and the controller will increase the network overhead. Therefore, we propose an online distributed algorithm (**Algorithm 1**) to achieve the transmitting group of data records.

Algorithm 1. Online Distributed Algorithm

```
1:  $c_d \leftarrow 0$, $c_c \leftarrow 0$
2:  while data processing not over do
3:    if confirmation message is received then
4:      remove and transmit confirmed packets from buffer
5:      begin buffering new packet
6:      $c_d \leftarrow 0$, $c_c \leftarrow 0$
7:    else if $c_d + c_c \geqslant c_t$ then
8:      send transmitting request
9:    else
10:     continue buffering packets
11:     recalculate $c_d$ and $c_c$
12:   end if
13: end while
```

We consider that multiple copies are launched for a task, and their intermediate outputs are sent to the corresponding task. Since task copies get the same input data, they generate the same output in the forms of key-value pairs. At the beginning of Algorithm 1, we define the variables c_d and c_c to indicate the delay cost of the data records in buffer and the communication cost between task duplication and the controller, respectively. Note that the algorithm is executed on each task duplication. There is a buffer to store the temporary data records. If the task duplication receives the confirmed message from the controller (line 3), it will remove the buffered packets and transmit packets to next phrase (line 4). Otherwise, it will check the sum of c_d and c_c, and then compare the sum with c_t. If the sum is larger than c_t, it will send the transmitting request to the controller (line 8). We set the condition (line 7) to buffer the produced data sets into one group. The reason is that our primary goal is to minimize network traffic, and therefore the extra cost cannot be larger than the network traffic cost.

6 Evaluation

We conduct extensive simulations to evaluate the performance of our proposed work with the following two operations.

- Only the controller manages the execution of task duplications to deliver the intermediate key - value pairs without overlap.
- Each task duplication executes the proposed algorithm to reduce the network overhead. To compare with our work, the random allocation algorithm is executed on each machine to deliver group of data records.

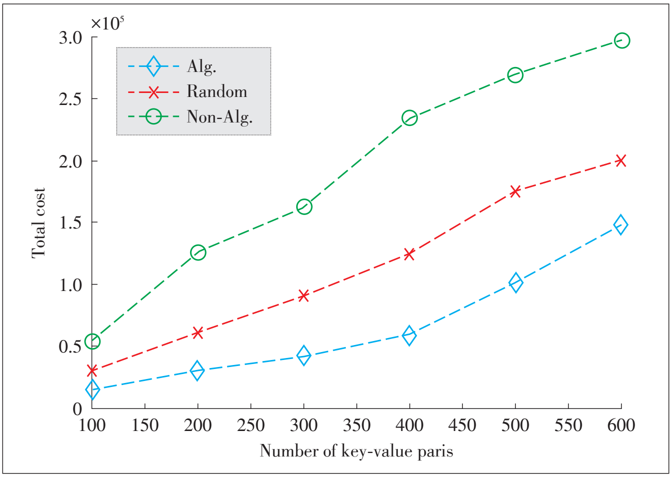
To our best knowledge, we propose to mitigate task stragglers and minimize total network cost for pipelining environment. The intermediate data records (*key/value pairs*) are associated with random size within [1–50], and each task has 30 task duplications in our simulations. The time of producing an intermediate data records is randomly set within [2–6] ms. For comparison, we also show the results of an random grouping algorithm and the one without traffic control.

We first evaluate the performance of our proposed algorithm by comparing with the random allocation algorithm and no group allocation algorithm. The controller chooses the machine that finishes its assigned task first, and then applies algorithms to divide the intermediate data into different groups. First of all, we tested intermediate key values' numbers of 100, 200, 300, 400, 500 and 600 for pipelining data processing output, and set the fixed task numbers to 30. As shown in **Fig. 3**, although the total cost increases as the number of key - value pairs increase for all the cases, our proposed algorithm has much lower increase than the other two schemes.

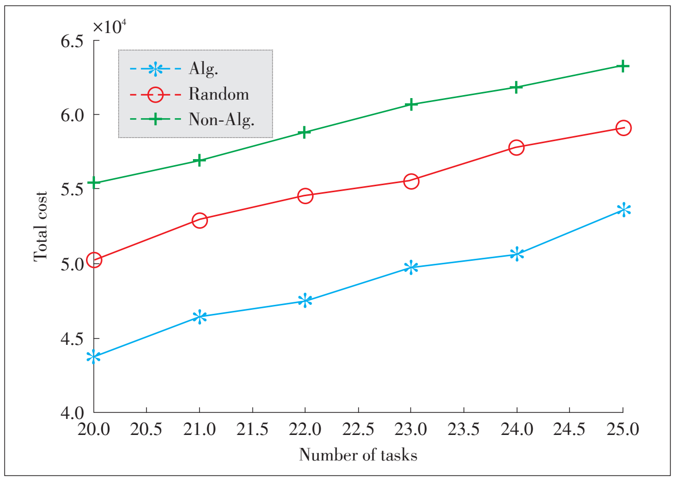
We then test the performance under the case of the fixed number of intermediate key value pairs, 300. As shown in **Fig. 4**, the total cost shows as an increasing function of number of tasks from 20 to 25 for all the cases. In particular, when the number of tasks is set to 25, the total cost of our proposed algorithm is about 5.3×10^4, while the total cost of no algorithm is 6.3×10^4, with a reduction of 25 percent. In contrast to the random allocation algorithm, our proposed algorithm also has a higher performance, because the random allocation algorithm does not consider the data delay.

As shown in **Fig. 5**, we study the influence of different number of task duplications by increasing the number of duplications form 10 to 15. The results show that our proposed algorithm always outperforms the other two schemes. That is because it requires only 2 times communication with the controller after delivering a group of data records, which is much smaller than the communication times without algorithm. Moreover, we make a tradeoff between the communication cost and the data delay cost and try to minimize the total cost. However, the random allocation algorithm does not take the data delay cost into consideration. Therefore, the total cost of the random allocation algorithm is much bigger than our proposed algorithm in the same cases.

Finally, we study the performance of three schemes under



▲**Figure 3. Total cost vs. the number of key-value pairs.**

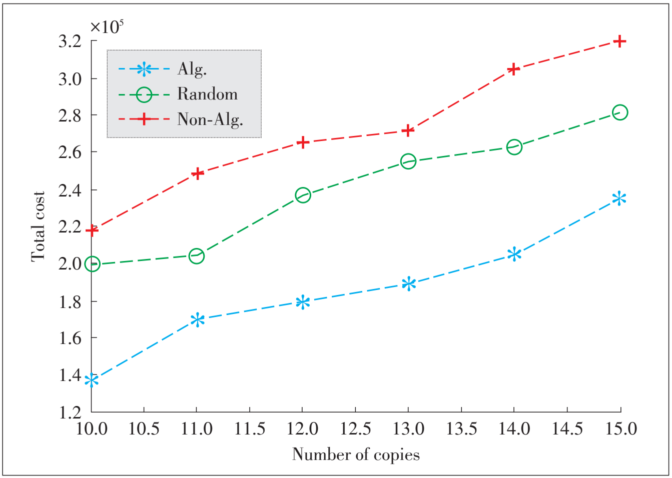


▲**Figure 4. Total cost vs. the number of tasks.**

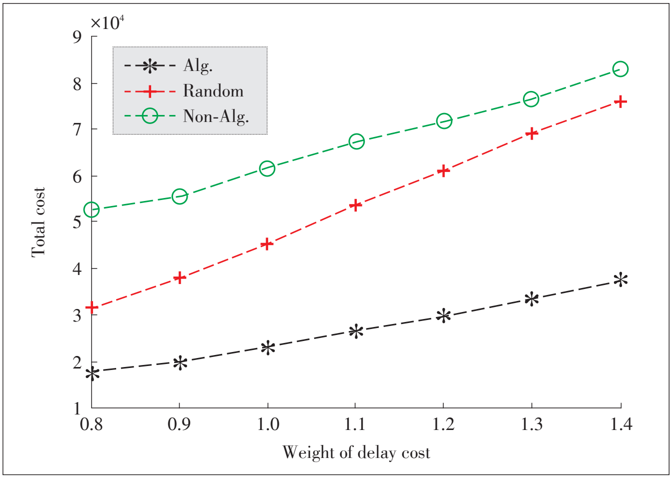
different weights of ratio by changing its value from 0.8 to 1.4. A small value of ratio indicates less importance of delay cost. From **Fig. 6**, we observe that the total cost of the random allocation algorithm has a sharp increase. However, the curve of our proposed algorithm increases slowly and it still outperforms the other two schemes.

7 Conclusions

In this paper, we study the principle of the parallel computing frameworks in cloud. We also use the technique of task duplication and strive for a traffic awareness for online big data processing. This approach can effectively accelerate job execution while avoiding redundant data transmission. To make a good tradeoff between network overhead and data delay, we design an online grouping algorithm to eliminate the traffic cost in cloud network. Finally, the performance of the proposed algorithm is evaluated by extensive simulations.



▲**Figure 5. Total cost vs. the number of task duplications.**



▲**Figure 6. Total cost vs. weight of delay.**

References

[1] S. Zou, X. Wen, K. Chen, et al., "Virtualknotter: online virtual machine shuffling for congestion resolving in virtualized datacenter," *Computer Networks*, vol. 67, pp. 141–153, 2014. doi: 10.1109/ICDCS.2012.25.

[2] G. Ananthanarayanan, A. Ghodsi, S. Shenker, and I. Stoica, "Effective straggler mitigation: attack of the clones," in *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI' 13)*, Lombard, USA, 2013, pp. 185–198.

[3] G. Ananthanarayanan, M. C.-C. Hung, X. Ren, et al., "Grass: trimming stragglers in approximation analytics," in *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI' 14)*, Seattle, USA, 2014, pp. 289–302.

[4] C.-Y Hong, S. Kandula, R. Mahajan, et al., "Achieving high utilization with software-driven WAN," in *SIGCOMM' 13*, Hong Kong, China, 2013, pp. 15–26. doi: 10.1145/2534169.2486012.

[5] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," in *Proc. 6th USENIX Symposium on Operating Systems Design and Implementation (OSDI' 04)*, San Francisco, USA, 2004, vol. 6, pp. 10–10.

[6] The Apache Software Foundation. (2017, Mar. 29). Apache Hadoop [Online]. Available: http://hadoop.apache.org

[7] T. Condie, N. Conway, P. Alvaro, et al., "Mapreduce online," in *Proc.7th USENIX Conference on Networked Systems Design and Implementation (NSDI' 10)*, San Jose, USA, 2010, pp. 21–21.

[8] G. D. Ghare and S. T. Leutenegger, "Improving speed up and response times by replicating parallel programs on a snow," in *11th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP)*, Cambridge, USA, 2005, pp. 264-287.

[9] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica, "Improving MapReduce performance in heterogeneous environments," in *Proc. 8th USENIX Symposium on Operating Systems Design and Implementation (OSDI' 08)*, San Diego, USA, 2008, pp. 29–42.

[10] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: distributed data-parallel programs from sequential building blocks," in *Proc. 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems*, Lisbon, Portugal, 2007, pp. 59–72. doi: 10.1145/1272996.1273005.

[11] G. Ananthanarayanan, S. Kandula, A. Greenberg, et al., "Reining in the outliers in map- reduce clusters using mantri," in *Proc. 9th USENIX Symposium on Operating Systems Design and Implementation (OSDI' 10)*, Vancouver, Canada, 2010, pp. 265–278.

[12] A. Baratloo, M. Karaul, Z. M. Kedem, and P. Wijckoff, "Char- lotte: metacomputing on the Web," *Future Generation Computer Systems*, vol. 15, no. 5–6, pp. 559–570, 1999.

[13] M. C. Rinard and P. C. Diniz, "Commutativity analysis: A new analysis framework for parallelizing compilers," in *Proc. ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, Philadelphia, USA, 1996, pp. 54–67.

[14] Q. Zang, H. Y. Chan, P. Li, and S. Guo, "Software-defined data shuffling for big data jobs with task duplication," in *45th International Conference on Parallel Processing Workshops (ICPPW)*, Philadelphia, USA, 2016, pp. 403– 407. doi: 10.1109/ICPPW.2016.62.

Manuscript received: 2017-06-23

Biographies

ZANG Qimeng (zangqm.uoa@gmail.com) is a graduate student in the department of Computer Science and Engineering, The University of Aizu, Japan. His research interests mainly include big data, cloud computing and RFID system.

GUO Song (song.guo@polyu.edu.hk) received his Ph.D. in computer science from University of Ottawa, Canada. He is currently a full professor at Department of Computing, The Hong Kong Polytechnic University (PolyU), China. Prior to joining PolyU, he was a full professor with The University of Aizu, Japan. His research interests are mainly in the areas of cloud and green computing, big data, wireless networks, and cyber-physical systems. He has published over 300 conference and journal papers in these areas and received multiple best paper awards from IEEE/ACM conferences. His research has been sponsored by JSPS, JST, MIC, NSF, NSFC, and industrial companies. Dr. GUO has served as an editor of several journals, including *IEEE TPDS*, *IEEE TETC*, *IEEE TGCN*, *IEEE Communications Magazine*, and *Wireless Networks*. He has been actively participating in international conferences serving as general chairs and TPC chairs. He is a senior member of IEEE, a senior member of ACM, and an IEEE Communications Society Distinguished Lecturer.

DOI: 10.3969/j.issn.1673-5188.2017.04.006
http://kns.cnki.net/kcms/detail/34.1294.TN.20171020.1814.002.html, published online October 20, 2017

Technical Analysis of Network Plug-in Flannel for Containers

YANG Yong, DONG Xiugang, and DONG Zhenjiang
(Nanjing R&D Center, ZTE Corporation, Nanjing 210012, China)

Abstract
The development of cloud computing has made container technology a hot research issue in recent years. The container technology provides a basic support for micro service architecture, while container networking plays an important role in application of the container technology. In this paper, we study the technical implementation of the Flannel module, a network plug-in for Docker containers, including its functions, implementation principle, utilization, and performance. The performance of Flannel in different modes is further tested and analyzed in real application scenarios.

Keywords
container technology; Docker; network plug-in; Flannel

1 Overview

With the development of cloud computing, container technology has become one of the hot research issues in recent years. In order to utilize container technology in the production environment, container network should be addressed firstly [1]. The container networking technology [2] connects inter-host containers and isolates each container network with others. Currently, the container networking technology has two solutions [3]: overlay networking and network routing. In overlay networking, application layer protocols are encapsulated and overlaid on the existing IP layer. Typical applications include Weave, Flannel, and Open vSwitch. In network routing, container networking is implemented through routing on network Layer 3 or 2. Typical applications include Calico and Macvlan. The overlay networking solution has no additional requirements on the existing network and container networking is available without the need to change the underlying network, so this mature solution is widely applied. However, with the increase of network nodes, the network is getting more and more complicated, and it is becoming more difficult and costly to maintain the network [4]. In addition, overlay networking is implemented through encapsulation on the application layer, so performance is also a large concern. Network routing features high network transmission performance and easy network maintenance, but has particular requirements on routers and switches in networking, which affects its scope of application.

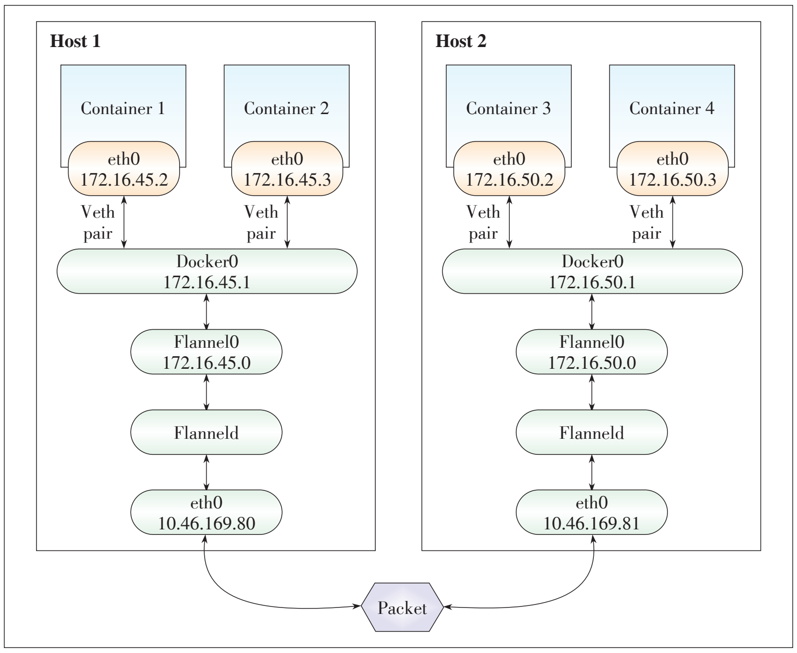
Flannel, a representative of the overlay networking solution for container networking, is a network plug-in service designed by CoreOS for Kubernetes. With Flannel, each Docker container created on different hosts in the container cluster can have a unique virtual IP address in the cluster [6], and Docker container networks can interwork with each other. Flannel is an independent open-source project using the Apache License Version 2.0 protocol and compiled with the Go language. Flannel is easy to use with a comprehensible implementation principle and has become a mainstream network plug-in recommended by many container solution providers.

This paper will provide an in-depth analysis of the implementation principle, as well as its utilization methods and modes of Flannel.

2 Implementation Principle of Flannel Network

Flannel provides an independent subnet for each host, and the network information of the entire cluster is stored on etcd, a distributed lock service. The IP address of a target container in the subnet of the host is queried from etcd for forwarding data across hosts. **Fig. 1** shows the implementation principle.

In Fig. 1, Host 1 and Host 2 are two hosts, on each of which two containers are operating. If the user datagram protocol (UDP) mode [7] is used for a request that needs to be sent from Container1 to Container3, the virtual network interface controller (NIC) Flannel0 (the virtual NIC name in virtual extensible



▲**Figure 1. Implementation principle of flannel network.**

LAN (VXLAN) mode may be different) is created.

The implementation procedure is described as follows:

1) IP datagrams are encapsulated and sent through eth0 of the container.
2) Container1's eth0 interacts with Docker0 through veth pair and sends a packet to Docker0. Docker0 then forwards the packet.
3) Docker0 determines that the IP address of Container3 points to an external container by querying the local routing table, and sends the packet to the virtual NIC Flannel0.
4) The packet received by Flannel0 is forwarded to the Flanneld process. The Flanneld process encapsulates the packet by querying the routing table maintained by etcd and sends the packet through eth0 of the host.
5) The packet determines the target host in the network across hosts.
6) The Flanneld process that listens to the 8285 port on the target host decapsulates the packet.
7) The decapsulated packet is forwarded to the virtual NIC Flannel0.
8) Flannel0 queries the routing table, decapsulates the packet, and sends the packet to Docker0.
9) Docker0 determines the target container and sends the packet to the target container.

3 Network Forwarding Modes Supported by Flannel

The Flannel configuration file is stored under the /coreos.com/network/config directory of etcd and can be viewed by running the etcdctl command. The directory where the configuration file is stored can be changed through --etcd-prefix.

The configuration file is a json file, consisting of the following five key parameters:

- Network (character string): IPv4 Classless Inter-Domain
  Routing (CIDR) of the entire Flannel network. It is the unique required keyword.
- SubnetLen (integer type): length of the subnet allocated. 24 is set as default (eg. /24).
- SubnetMin (character string): start IP address allocated to a subnet. The first address of the subnet is set as default.
- SubnetMax (character string): end IP address allocated to a subnet. The last address of the subnet is set as default.
- Backend (dictionary): type and specific configuration used by the back end. The keywords supported in the dictionary are described below (default: UDP).

Backend specifies the transfer mode of an encapsulated packet and supports the following modes:

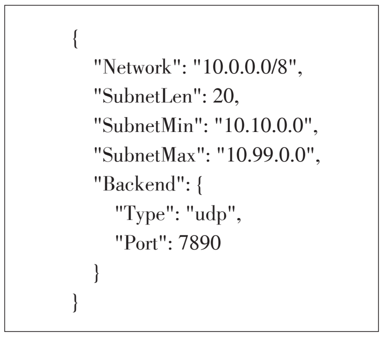
1) UDP: encapsulating packets through UDP [8]
- type (character string): udp
- Port (digit): UDP port number used for sending encapsulated packets, with 8285 set as default

2) vxlan: encapsulating packets through kernel VXLAN [9]
- type (character string): vxlan
- VNI (digit): ID used by the VXLAN [10] (VNI), with 1 set as default
- Port (digit): UDP port number used for sending encapsulated packets, with 8472 set as default (kernel default value)

3) host-gw: creating an IP route to a subnet through the IP address of the remote host. This requires the hosts that are operating Flannel are directly interconnected on Layer 2.
- type (character string): host-gw

4) aws-vpc: used for the containers operating on AWS
5) gce: used for containers operating on GCE
6) alloc: only implementing network distribution, with no packet forwarding.

Fig. 2 shows a json file for packet forwarding in UDP mode. The technical implementations in UDP, VXLAN, and host-



```
{
    "Network": "10.0.0.0/8",
    "SubnetLen": 20,
    "SubnetMin": "10.10.0.0",
    "SubnetMax": "10.99.0.0",
    "Backend": {
        "Type": "udp",
        "Port": 7890
    }
}
```

Figure 2. ▶
A json file in UDP mode for sending packets.

GW modes are described below.

4 Implementation Principle and Analysis of Flannel Network Forwarding Modes

Flannel provides the networking service for containers and uses etcd to store the network configuration information [11], [12]. Flannel obtains subnet information from etcd, declares the network segment of its subnet, and stores it in etcd. Before data forwarding across hosts, Flannel queries the IP address of the host corresponding to the subnet and sends data to flanneld of the host, and the flanneld forwards the data. The following analyzes the data stream sending and receiving flows in UDP, VXLAN, and host-GW modes.

4.1 UDP Mode

UDP is the default mode used by Flanneld. The virtual NIC flannel0 is added on a host and uses the 8285 as the listening port by default.

According to the Wireshark analysis (**Fig. 3**), the data is forwarded in the UDP mode. The source and destination IP addresses of the source packet are the IP address of the host, and the IP addresses of containers are encapsulated into the packet.

4.2 VXLAN Mode

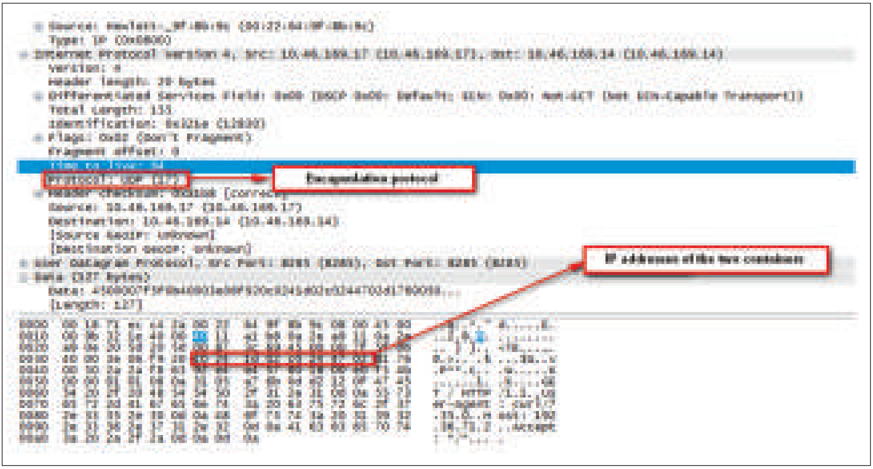
In VXLAN mode, Flannel runs a flanneld process on each host and creates a VXLAN device (virtual NIC Flannel.x, where x indicates the VNI). In this test, VNI is 1, so the virtual NIC Flannel.1 is created.

According to the Wireshark analysis as shown in **Fig. 4**, the data is forwarded also through UDP, and the source and destination IP addresses are the IP addresses of the host. Some VXLAN encapsulation protocols are added in the packet (which is different from the UDP mode), and the IP addresses of the two containers are encapsulated into the packet.

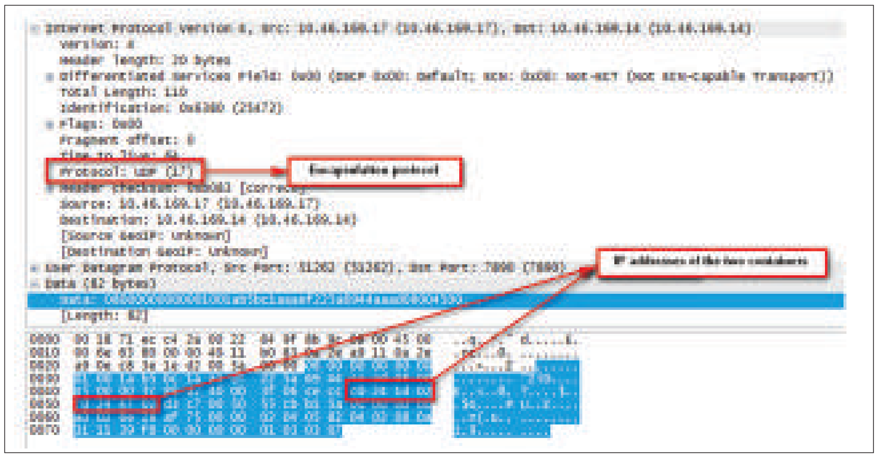
4.3 Host-GW Mode

Different from that in UDP and VXLAN modes, no Flannel virtual NIC is created in the host-GW mode, but the hosts need to be interconnected on Layer 2. In this test, the IP addresses of two containers are 192.38.88.2 and 192.38.95.2.

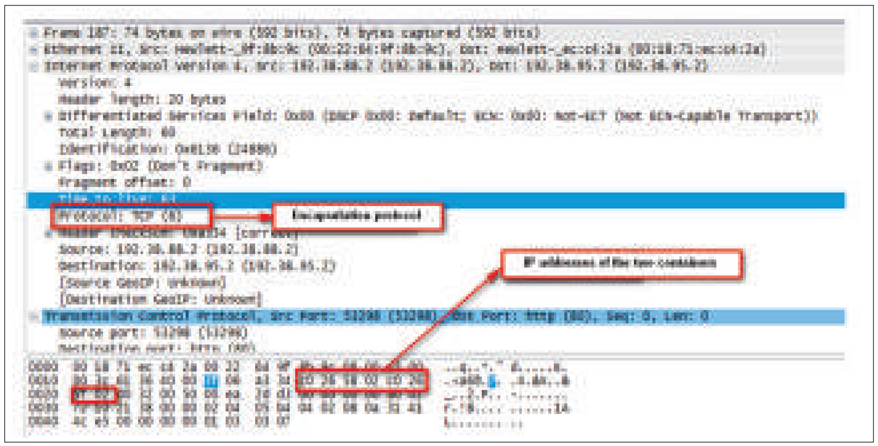
According to the Wireshark analysis (**Fig. 5**), the source and destination IP addresses in the packet are the IP addresses of the two containers but not the IP address of the host. No UDP-based packet encapsulation is implemented. Instead, the pack-


```
⊞ Source: Hewlett-_9f:8b:9c (00:22:64:9f:8b:9c)
   Type: IP (0x0800)
⊟ Internet Protocol Version 4, Src: 10.46.169.17 (10.46.169.17), Dst: 10.46.169.14 (10.46.169.14)
   Version: 4
   Header length: 20 bytes
 ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
   Total Length: 155
   Identification: 0x32be (12990)
 ⊞ Flags: 0x02 (Don't Fragment)
   Fragment offset: 0
   Time to live: 64
   Protocol: UDP (17)
 ⊞ Header checksum: [illegible] [correct]
   Source: 10.46.169.17 (10.46.169.17)
   Destination: 10.46.169.14 (10.46.169.14)
   [Source GeoIP: Unknown]
   [Destination GeoIP: Unknown]
⊞ User Datagram Protocol, Src Port: 8285 (8285), Dst Port: 8285 (8285)
⊟ Data (127 bytes)
   Data: 4500007f3f0b40003e06f920c8245d02c02447024170005b...
   [Length: 127]
0000 [illegible]
```

▲Figure 3. Wireshark analysis for UDP mode.


```
⊟ Internet Protocol Version 4, Src: 10.46.169.17 (10.46.169.17), Dst: 10.46.169.14 (10.46.169.14)
   Version: 4
   Header length: 20 bytes
 ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
   Total Length: 110
   Identification: 0x6380 (25472)
 ⊞ Flags: 0x00
   Fragment offset: 0
   Time to live: 64
   Protocol: UDP (17)
 ⊞ Header checksum: [illegible] [correct]
   Source: 10.46.169.17 (10.46.169.17)
   Destination: 10.46.169.14 (10.46.169.14)
   [Source GeoIP: Unknown]
   [Destination GeoIP: Unknown]
⊞ User Datagram Protocol, Src Port: 51262 (51262), Dst Port: 7890 (7890)
⊟ Data (82 bytes)
   Data: [illegible]
   [Length: 82]
0000  00 18 71 ec c4 2a 00 22  64 9f 8b 9c 08 00 45 00
0010  00 6e 63 80 00 00 40 11  [illegible]
[illegible]
0070  01 11 39 f8 00 00 00 00  01 01 03 [illegible]
```

▲Figure 4. Wireshark analysis for VXLAN mode.


```
⊞ Frame 187: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
⊞ Ethernet II, Src: Hewlett-_9f:8b:9c (00:22:64:9f:8b:9c), Dst: Hewlett-_ec:c4:2a (00:18:71:ec:c4:2a)
⊟ Internet Protocol Version 4, Src: 192.38.88.2 (192.38.88.2), Dst: 192.38.95.2 (192.38.95.2)
   Version: 4
   Header length: 20 bytes
 ⊞ Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00: Not-ECT (Not ECN-Capable Transport))
   Total Length: 60
   Identification: 0x8138 (24888)
 ⊞ Flags: 0x02 (Don't Fragment)
   Fragment offset: 0
   Time to live: 63
   Protocol: TCP (6)
 ⊞ Header checksum: [illegible] [correct]
   Source: 192.38.88.2 (192.38.88.2)
   Destination: 192.38.95.2 (192.38.95.2)
   [Source GeoIP: Unknown]
   [Destination GeoIP: Unknown]
⊟ Transmission Control Protocol, Src Port: 53298 (53298), Dst Port: http (80), Seq: 0, Len: 0
   Source port: 53298 (53298)
   Destination port: http (80)
0000  00 18 71 ec c4 2a 00 22  64 9f [illegible]
0010  00 3c 81 38 40 00 3f 06  a1 34 c0 26 58 02 c0 26
0020  5f 02 [illegible]
0030  [illegible]                04 02 08 0a 31 41
0040  4c e5 00 00 00 00 01 03  03 07
```

▲Figure 5. Wireshark analysis for host-GW mode.

et is transmitted through TCP.

5 Conclusions

Our tests were conducted in two servers with the same configuration, including the servers with 100M network card, operating system ubuntu14.04, Flannel v0.7.0, and etcd v2.3.1. Besides, Iperf, the network performance testing tool for testing the maximum bandwidth performance of TCP or UDP, is used, and two containers are running on each of the hosts. **Table 1** describes the results of the performance tests.

▼Table 1. Performance comparision of direct data transmission between two hosts with UDP/VXLAN/host-GW-based transmission through Flannel.

Mode	Rate (M/s)	Description
Direct transmission between two hosts	94.3	There is a normal performance loss in direct transmission between two hosts.
UDP	92.7	There is greater performance loss.
VXLAN	91.2	
Host-GW	94.3	The performance is approximate to that of direct transmission between two hosts.

GW: gateway
UDP: user datagram protocol
VXLAN: virtual extensible local area network

According to the testing results, the host-GW mode provides the best performance, because no virtual NIC is created in this mode, which thus reduces the number of forwarding operations and improves network performance. However, in this mode, the destination address is the container's IP address and media access control (MAC) addressing is used, which requires that the two hosts be interconnected on Layer 2.

References

[1] J. Turnbull, *The Docker Book: Containerization is the New Virtualization*. Seattle, USA: Amazon Digital Services, Inc., 2014.
[2] dotCloud. (2017, Apr. 30). Docker: build, ship and run any app, anywhere [Online]. Available: https://www.docker.com
[3] B. Yang, W. Dai, and Y. Cao, *Docker Primer (in Chinese)*. Beijing, China: China Machine Press, 2014.
[4] H. Sun Hongliang, *The Source Code Analysis of Docker (in Chinese)*. Beijing, China: China Machine Press, 2015.
[5] K. Wang, G. Zhang, and X. Zhou, "Research on the container-based virtualization technology," *Computer Technology and Development*. vol. 25, no. 8, pp. 138–141, Aug. 2015. doi: 10.3969/ j.issn.673-629X.2015.08.029.
[6] S. Du, "The application of virtual reality technology in computer network platform," *Computer Engineering & Software*, vol. 34, no. 1, pp. 45–46, Jan. 2013.
[7] G. Peng, "UDP-based point-to-point fast and reliable transmission model," Dissertation, Sun Yat-Sen University, Guangzhou, China, 2013.
[8] F. Zhao and Z. Ye, "Contract analysis of UDP and TCP and improvement of UDP in reliability," *Computer Technology and Development*, vol. 16, no. 9, pp. 219–221, Sept. 2006.
[9] H. Zhao, Y. Xie, and F. Shi, "Network Virtualization and Network Function Virtualization," *ZTE Technology Journal*, vol. 20, no. 3, pp. 8–11, Jun. 2014. doi: 10.3969/j.issn.1009-6868.2014.03.002.
[10] Z. Lu, Z. Jiang, and B. Liu, "A VXLAN-Based Virtual Network Access Control Method," *Computer Engineering*, vol. 40, no. 8, pp. 86–90, Aug. 2014.
[11] L. Wu Longhui, *Kubernetes*. Beijing, China: Publishing House of Electronics Industry, 2016.
[12] SEL of Zhejiang University, *Docker Container and Container Cloud*. Beijing, China: Posts & Telecom Press, 2015.

Manuscript received: 2017-07-26

Biographies

YANG Yong (yang.yong3@zte.com.cn) works with ZTE Corporation and focuses on ICT application R&D and software architecture technology research. His research interests include service platforms, capability opening, multimedia technologies, and cloud computing.

DONG Xiugang (dong.xiugang20@zte.com.cn) works with ZTE Corporation. His research interests include the Web and container technologies (including the micro service and container cluster management technologies) and their application, and natural language processing.

DONG Zhenjiang (dong.zhenjiang@zte.com.cn) works with ZTE Corporation. His research interests include security technologies , cloud computing and artificial intelligence.

DOI: 10.3969/j.issn.1673-5188.2017.04.007
http://kns.cnki.net/kcms/detail/34.1294.tn.20171031.1653.002.html, published online October 31, 2017

Virtualization Technology in Cloud Computing Based Radio Access Networks: A Primer

ZHANG Xian and PENG Mugen
(Beijing University of Posts and Telecommunications, Beijing 100876, China)

Abstract

Since virtualization technology enables the abstraction and sharing of resources in a flexible management way, the overall expenses of network deployment can be significantly reduced. Therefore, the technology has been widely applied in the core network. With the tremendous growth in mobile traffic and services, it is natural to extend virtualization technology to the cloud computing based radio access networks (CC-RANs) for achieving high spectral efficiency with low cost. In this paper, the virtualization technologies in CC-RANs are surveyed, including the system architecture, key enabling techniques, challenges, and open issues. The enabling key technologies for virtualization in CC-RANs mainly including virtual resource allocation, radio access network (RAN) slicing, mobility management, and social-awareness have been comprehensively surveyed to satisfy the isolation, customization and high-efficiency utilization of radio resources. The challenges and open issues mainly focus on virtualization levels for CC-RANs, signaling design for CC-RAN virtualization, performance analysis for CC-RAN virtualization, and network security for virtualized CC-RANs.

Keywords

network virtualization; CC-RAN; RAN slicing; fog computing

1 Introduction

With the explosive growth of wireless data traffic and differentiated performance requirements of mobile services and the Internet of Things (IoT), traditional networks are facing serious challenges in satisfying capacity demands, energy efficiency, and flexibility requirements [1]. Meanwhile, especially from 3G to 4G, many wireless networks have been deployed to satisfy data traffic demands and guarantee seamless coverage [2]. However, those heterogeneous wireless networks usually overlap the same areas, thus compromising the full use of infrastructure resources and radio resources [3]. In addition, based on the dedicated hardware and vertical network architecture, the wireless networks are designed for peak capacity demands and operated separately, with wireless resources and network resources being unable to be dynamically allocated. Those problems and challenges cause the capital expenses (CAPEX) and operation expenses (OPEX) not cost-efficient. From the perspective of network resources, 5G wireless communication networks are expected to intelligently integrate all kinds of resources from multiple resource owners to provide augmented and data-intensive services in a multi-vendor multi-proprietor scenario with maximizing resource utilization [4]. Traditional networks based on dedicated hardware and other resources with static or periodic allocation are unable to satisfy the requirements and facing great challenges. However, with virtualization, which can pool the resources as “virtual machines (VMs)” to realize dynamic sharing and reprogramming, the network functions and resources can be managed flexibly with higher utilization. Under different network environment, virtualization technology has different flexibility and complexity as well as challenges.

With the goal of creating a more flexible, cost-effective, and open wireless communication network, some ideas and concrete initiatives of the virtualization-driven 5G have been discussed on the 34th meeting of Wireless World Research Forum (WWRF34) [5]. Virtualization has been seen as one of the main evolution trends in the forthcoming 5G cellular networks [6]. Software defined networking (SDN), network function virtualization (NFV) and cloud computing are considered as the promising technologies to realize virtual networks, especially in core networks (CNs) and network controls. They are widely treated as the important virtualization technology in ICT area. Concretely, the fundamental principle in SDN is flexibly decoupling the control plane (i.e., configuration and management) and data plane (i.e., forwarding), therefore, SDN has been regarded as a crucial driver to virtualize wireless access and core networks [7]. NFV is mainly focused on functionality modular design and general-purpose platform replacing dedicated hardware. Through migrating network functions from the dedicated hardware to general-purpose networking and computing platforms, the cost of deploying and operating infrastructures can be reduced [8]. Through cloud computing, distributed computation infrastructures and services can also be quickly deployed, dynamically managed, and globally optimized [9].

With tremendous growth in wireless traffic and services, virtualization has been introduced into wireless networks, such as mobile cellular network virtualization [10] and software-defined wireless mesh networks [11]. At the same time, as virtualization technology has been widely applied in CNs, in order to achieve its full potential and relief the pressure of RANs, it is necessary to extend virtualization technology to the radio access segment [8]. However, the inherent nature of wireless communications, such as broadcasting characteristic, stochastic fluctuation of wireless channel quality, severe interference and computational complexity, lets the traditional network architecture and resource allocation for high system capacity and data rates unable to satisfy diverse services with differentiated performance requirements. Fortunately, considering virtualization into RAN environments, resource virtualization with centralized and hierarchical control and management can provide more elaborate resource granularity and a greater degree of resource sharing. In addition, the signaling overhead and complexity can be decreased because of centralized processing gain and signaling-data split.

However, from the perspective of current related studies, network virtualization and wireless network virtualization are mainly focusing on mobile network virtualization based SDN and NFV. In [12], from the data plane and control plane perspective, a virtual server platform—NetVM was proposed for network functions, based on which the SDN NFV (SDNFV) for controlling the smarter data plane was introduced. The joint design of software-defined wireless networking (SDWN) and wireless network virtualization (WNV) was presented for addressing the crucial challenges in future networks and the NFV resource allocation (NFV-RA) problems were discussed in [13], while the network virtualization and resource description in SDWN were discussed in [14]. In addition, according to the requirements of 5G, a concrete approach for wireless network virtualization, including the framework model, control schemes were mentioned in [15]. The SDWN was discussed in [16], including use cases, generic architecture detailing in terms of modules, interfaces, and high-level signaling. In [17], given that SDNs mainly adopt forwarding rules as the basic control unit to manage network traffics, the rules increase dramatically after network virtualization. If the rule space is not large enough for processing new packets, network devices have to communicate to the SDN controller, which leads to latency for users and heavy load to controller. Therefore, the amount of social IoT groups vRANs under limited rule space and latency requirement has been optimized.

As to RAN virtualization, the current existing studies are mainly on software-defined and virtualized wireless access, concretely including the flow oriented perspective, protocol oriented perspective, and spectrum oriented perspective [18]. Furthermore, from the flow oriented perspective, the authors of [19] and [20] researched the network virtualization substrate (NVS) implemented in base stations, which mainly focused on the management, scheduling, and service differentiation of different data flows among different slices. In [21], multiple CN operators have accessed to a common RAN, which needs RAN virtualization to map virtual network elements onto wireless resources of the existing physical network. In [7], multiple isolated virtual networks are built on one or more physical network substrates and they are able to use customized network protocols, signal processing and network management functionalities to satisfy the intended services. Both [7] and [17] can be regarded as multiple wireless protocol instances that are isolated, customized and managed on the same wireless hardware from the protocol oriented perspective. As to the spectrum oriented perspective, the resources to be sliced are radio frequency (RF) bands and white spectrum [18] for extended sharing and high radio resource utilization, which can be treated as the spectrum virtualization.

To improve the spectral efficiency and energy efficiency, many kinds of cloud computing based RANs (CC-RANs) have been proposed, such as the cloud RAN (C-RAN), heterogeneous C-RAN (H-CRAN), and even fog RAN (F-RAN). Unfortunately, the virtualization technology in CC-RANs has not been widely and systematically researched, which can be a promising direction for future advanced RAN design. CC-RAN virtualization can fit well the type and amount, as well as the granularity of resources for the service performance requirements by dynamical and flexible abstraction and slicing.

The remainder of this paper is organized as follows. we provide a brief survey on the background of network virtualization in Section 2, including the related definitions, projects and business models, motivations, and contributions. In Section 3, we introduce the components of network virtualization with related realizing technologies. Three main CC-RAN architectures with virtualization are comprehensively analyzed in Section 4. Following, we discuss the key enabling technologies for CC-RAN virtualization to satisfy the requirements of isolation, customization and high-efficiency utilization of radio resources in Section 5. In Section 6, challenges and open issues has been discussed. In Section 7, we concluded the paper.

2 Background of Network Virtualization

In this section, we present a brief review of history and development of network virtualization. The related definitions are also given and compared, especially on CC-RAN virtualization. Then we investigate the main projects on network virtualization and realted business models, especially for the virtualization in CC-RANs. Finally, we give a brief summary of the existing related articles and summarize our contributions.

2.1 History Development of Network Virtualization

Virtualization has accelerated the development of information technology. In an information and communications technology (ICT) system, virtualization can realize the physical hardware, such as servers, network and storage devices in a software form, which simplifies IT management and enables agile IT services delivery with low costs. With a lot of benefits, virtualization has gradually become a popular concept in ICT area, such as virtual machines, virtual memory, virtual data centers and even virtual network functions. In wired networks, virtualization has been applied for decades. The authors of [22] described virtual local area networks (VLANs), virtual private networks (VPNs), active and programmable networks, and overlay networks. The common feature of them is that the wired network virtualization is limited to one or two layers, thus the benefits of virtualization have not been fully exploited. In order to take full advantages of virtualization, the networks need to be fully virtualized, and services be completely separated from their underlying infrastructure.

Since the introduction of SDN and NFV, the network virtualization has been widely researched. In [23], a network function virtualization proof of concept (PoC) was reported, which demonstrated that the dynamic SDN control coordinating with a cloud management approach could bring added value to telecommunication operators and service providers. The PoC also demonstrated how telecommunication services based on NFV can be made self-adaptive to the network conditional and user's changing requirements. Based on modular design, NFV combing with SDN control can satisfy well the differentiated scenarios and performance demands by CN slicing and further end-to-end network slicing. Accordingly, users can get better QoS because of customization. Network virtualization has been considered as one of the most significant technologies for the future Internet. With increasing mobile Internet and IoT wireless data traffic and services, wireless network virtualization has been widely discussed.

2.2 Definitions of Network Virtualization

In order to avoid confusion, in this part, the related definitions on network virtualization in the recent literature have been summarized, including virtualization, NFV, network virtualization, WNV, and RAN virtualization.

2.2.1 Virtualization

Virtualization is the process of creating a software-based (or virtual) representation of something rather than a physical one, and the heart of virtualization is the "VM", a tightly isolated software container with an operating system and application inside [24]. More importantly, a thin layer of software in platforms called hypervisor decouples VM from the host, and the hypervisor can dynamically allocates computing resources to each VM as needed. Virtualization can be applied to applications, servers, storage, and networks, which is the most effective way to reduce IT expenses while boosting efficiency and agility for all size businesses.

2.2.2 Network Function Virtualization

NFV was proposed by the European Telecommunications Standards Institute (ETSI) in October 2012, for minimizing or even eliminating the dependence on proprietary hardware [25]. Taking advantages of the evolution of IT virtualization, NFV is transferring network functions from dedicated hardware appliances to software-based applications running on commercial off-the-shelf (COTS) equipment [26], such as high-volume servers, switches, and storage. Through NFV, network functions can be instantiated in various locations such as data centers, network nodes, and even end-user equipment as the network requires. The NFV framework, state-of-the-art, and implementation as well as challenges for next generation mobile network can be found in [25].

2.2.3 Network Virtualization

Network virtualization can be regarded as a process of sharing the entire network system with an optimized approach [15]. In [27], network virtualization has been defined from the perspective of network resources. Network virtualization is any form of partitioning or combining a set of network resources, and abstracting it to users such that each user, through its set of the partitioned or combined resources, has a unique, separate view of the network. The resources can be fundamental (nodes, links) or derived (topologies), which can be virtualized recursively. The node and link virtualization involves resource partition, combination, and abstraction. NFV is a strong technology candidate toward network virtualization. Through NFV, network functionalities can be encapsulated into software packages that can be distributed through the network and performed in a homogeneous environment [28], which will simply the implementation of network virtualization.

2.2.4 Wireless Network Virtualization

WNV has a very broad scope ranging from infrastructure virtualization, spectrum sharing, to air interface virtualization [22], which can be easily understood as the network virtualization emphasizing on mobile networks, including mobile CNs and RANs. Wireless network virtualization as the technologies in which physical wireless network infrastructure resources and physical radio resources can be abstracted and sliced into virtual wireless network resources holding certain corresponding functionalities, and shared by multiple parties through isolating each other [10], [22], [29]. In [30], based on wireless network virtualization technology, several concurrent virtual networks could run on the shared wireless physical substrate and the physical nodes as well as physical links are virtualized into several virtual nodes and virtual links belonging to different virtual networks. Recently, the research of network virtualization mainly concentrates on WNV.

2.2.5 RAN Virtualization

To enable end-to-end network virtualization, both the wireless CNs and RANs have to be virtualized [19]. The network virtualization has been focused on the design of network substrate to support multiple virtual networks. However, with the relatively mature application of the virtualization technology such as NFV and SDN in mobile CNs, recent research attention has shifted to meeting the baseband-processing requirements of RANs on high-volume IT hardware as well as centralized resource virtualization and dynamical allocation to avoid over-provisioning [31].

Through radio spectrum virtualization, infrastructure sharing, virtualization of multiple radio access technologies (RATs), and virtualization of computing resources, virtualization can be applied in the context of RANs [31]. In [32], RAN virtualization is explained as leveraging SDN/NFV to virtualize a portion of the RAN functionalities onto standard IT or COTS hardware in a central location or in the cloud, which has the potential to offer advantages such as smaller footprint and energy consumption through dynamic load balancing and traffic steering. However, the radio resource virtualization and sharing were not taken into full consideration in [32].

The similar definitions of RAN virtualization, such as wireless access virtualization in [7], software-defined and virtualized wireless access in [18], RAN sharing in [19], and software-defined radio (SDR) and software-defined fronthaul (SDF) networks in [33] also have been discussed in the recent literature.

From the related definitions above, RAN virtualization is included in the wireless network virtualization, and wireless network virtualization is included in network virtualization. All three of them generally take virtualization technology, such as NFV, as the enablers. However, RAN virtualization has to deal with problems specific to the characteristics of wireless access links that usually have more dynamic sets of users, user mobility, and varying channel conditions, which makes it harder to virtualize the wireless resources across multiple entities [19]. Therefore, there exist several challenges in RAN virtualization:

- A virtualized RAN (or a RAN slice) should be able to use a permitted amount of resources by dynamical allocating, which easily varies with network load and is influenced by interference, user mobility and operator policies.
- It is harder to achieve the two conflicting goals of isolation and efficient resource utilization across slices, since the RAN has to consider the resource sharing for uplink traffic and the up-down direction of resource allocation [20].
- Wireless networks often incur considerable overheads due to signaling and retransmissions; however, the resources are rather limited.

In addition, unlike virtualization of CNs, where the virtualized functionalities are executed by VM under the control of a hypervisor on centralized general purpose processors (GPPs), RAN virtualization can only run before the RAN actually needs to interface with the physical network by transmitting an RF signal [32], which needs the system to respond in real time to the RF signal. In order to achieve the full potential of RAN virtualization, the definition of RAN virtualization needs to be extended wider than that of virtualization in the IT area, such as resource abstraction and sharing. In this paper, the virtualization technology in CC-RANs is comprehensively surveyed.

2.3 Projects and Business Models

Network virtualization has been one of the hottest topics in telecommunications. However, when it comes to RANs, it may seem technically complex, risky and hard to deploy. The authors of [22] have given a summary on network virtualization projects; however, those projects mainly focus on wireless network virtualization. In order to investigate the newest research and achievements, it's necessary to review the latest projects on RAN virtualization as well as business models.

2.3.1 Projects

Recently, the projects on virtualization of RANs have been launched, including the Flexible Architecture for Virtualizable Future Wireless Internet Access (FLAVIA) [19], [34], Small Cell Forum [35], SDN at the Edges [23], 5G Exchange (5GEx) project [36], and Small cEllS coordinAtion for Multi-tenancy and Edge services (SEAME) project [37], etc.

FLAVIA is a European Union 7th Framework Programme (FP7) project that fosters a paradigm shift towards the future wireless Internet: from pre-designed link services to programmable link processors. The significant concept of FLAVIA is to expose flexible programmable interfaces to enable service customization and performance optimization via software-based exploitation of low-level operations and control, transmission timing, frame customization and processing, spectrum and channel management, power control, etc.

SDN at the Edges is the framework of activity "SDN at the Edges" founded in 2015 by the EIT-Digital initiative under the "Future Networking Solutions" action line, which aims at investigating how to create the technical conditions for accelerating the practical development of SDN and NFV solutions.

Small Cell Forum provided a comprehensive analysis on small cell virtualization through its operator group in June 2014. The report [35] describes the findings of that activity, in particular, the benefits of centralization and virtualization of small cell RAN, including improved coordination, enhanced scalability, reduced cost, accelerated upgrade lifecycle, and flexibility. In addition, the different functional splits and related performance benefits and constraints are also discussed.

5GEx Project, through designing an agile exchange mechanism for contracting, invoking, and settling the wholesale consumption of resources and virtual network services, aims to enable cross-domain orchestration of services over multiple administrations or over one multi-domain administration and that the services can be provisioned in less than 90 minutes.

SEAME, by bringing virtualization, control and intelligence to the network edge, proposes the Cloud-Enabled Small Cell (CESC) concept, a new multi-operator enabled small cell that integrates a virtualized execution platform for deploying virtual network functions (NVFs), supporting powerful self-management and executing novel applications and services inside the access network infrastructure.

2.3.2 Business Models

With abundant services and applications unprecedentedly appearing, mobile network operators (MNOs) are under great stress because of being pipelined with little profits. Since MNOs are looking for new approaches to be cost-efficient and to relieve fast increasing financial investment burden, new business models based on an extension to existing standardized network sharing functionality have been provided in [19], including on-demand capacity, wholesale-only network, and over-the-top service providers.

Comparing with the business models developing for network sharing above, in the network virtualization environment, according to which party physical resources and virtual resources belong to, business models can be described as the roles (the business models themselves), the functions of the roles, and the relation between the roles [22], [38]. A two-level model and a three-level model as well as the functions of roles, including infrastructures provider (InP), MNO, mobile virtual network operators (MVNOs), and service providers (SPs) have been concretely explained in [22]. In [38], the business models of network slicing as a service (NSaaS) have been classified into three classes, including business to business (B2B), business to consumer (B2C), and business to business to consumer (B2B2C). In addition, according to different control levels on mobile network resources, MVNOs are further classified into three types, including resellers, services providers, and full MVNOs [10]. The detail functions of each roles and differences as well as typical representatives are discussed in [22] and [38], thus omitting them here for brevity.

Finally, the business models can also be summarized as "something" as a service (XaaS). Infrastructure as a service (IaaS) and network as a service (NaaS) are respectively provided by InPs and MVNOs. SPs can also provide software as a service (SaaS) and cloud as a service (CaaS). An example might be RAN-as-a-Service (RANaaS), where RAN is offered like a cloud-service [39].

2.4 Motivations and Contributions

Recently, RAN virtualization has been gradually concentrated on in literature. However, on the one hand, the related definitions may be in a muddle and the existing literature on network virtualization usually lacks consideration on concrete RAN architectures. RAN virtualization can be unified in different RATs, each of which is dedicated to different services and offering a different quality of service (QoS), since virtualization can simplify the management of different RATs [31]. On the other hand, the current network virtualization and wireless network virtualization are mainly emphasizing on mobile CN virtualization based on SDN and NFV, and jointly designing the generic network virtualization architecture.

To provide a better understanding of research opportunities and challenges on virtualization technology in CC-RANs, the survey gives the state-of-the-art CC-RAN virtualization and wants to attract more attention on it, unleashing the potential gains given by CC-RAN virtualization. The main contributions of the paper can be summarized as follows.

We systemically review the related definitions, and based on a brief investigation on the current network virtualization research, we give the components of network virtualization.

In order to take different advantages of CC-RANs and to satisfy the different performance requirements, we illustrate and classify the virtualization architecture into three trendy CC-RANs.

In order to satisfy the requirements of virtual networks, including isolation, customization and high radio resource utilization of CC-RAN virtualization, we review the key enabling technologies. Finally, we discuss the challenges and open issues as the broad perspectives in CC-RAN virtualization.

3 Components of Network Virtualization

Network virtualization can be treated as the process of virtualizing a set of network resources. Since RAN virtualization is virtualization technology applied in RAN context, the components of network virtualization are same for RAN virtualization. From the perspective of resources, network virtualization can be treated as abstracting and dynamically allocating resources to efficiently share by multiple virtual networks via isolations. In **Fig. 1**, from the perspective of resource, the network virtualization architecture consists of three layers, including the infrastructure and wireless resources layer (L1), virtual resources layer (L2), and logical networks and services layer (L3). Through virtualization technology, the physical resources in L1 can be abstracted as virtual resources, which can be dynamically allocated to L3 and orchestrated for service slices.

3.1 Infrastructure Virtualization

The legacy network deployments are based on physical middle-boxes and vendor-locked equipment [23]. It may be impossible to achieve the scalability and flexibility in 5G network with higher resource utilization and satisfying various service demands. Meanwhile, through purchasing or developing software and running it on physical machines such as commodity services, all kinds of network functions can be realized. However, the gains of flexibility, dynamic resource scaling, and energy efficiency will be declined.

On the one hand, the physical network elements, such as antennas, BSs, processor hardware and routers, can be virtualized to support sharing of multiple operators by infrastructure virtualization [22]. Hardware and network sharing is beneficial for small cells in order to avoid massive over-provisioning [31], especially for the heterogeneous and dense deployments in 5G networks. In [40], a novel concept of Universal Intelligent Small Cell (UnISCell) has been proposed for enabling the

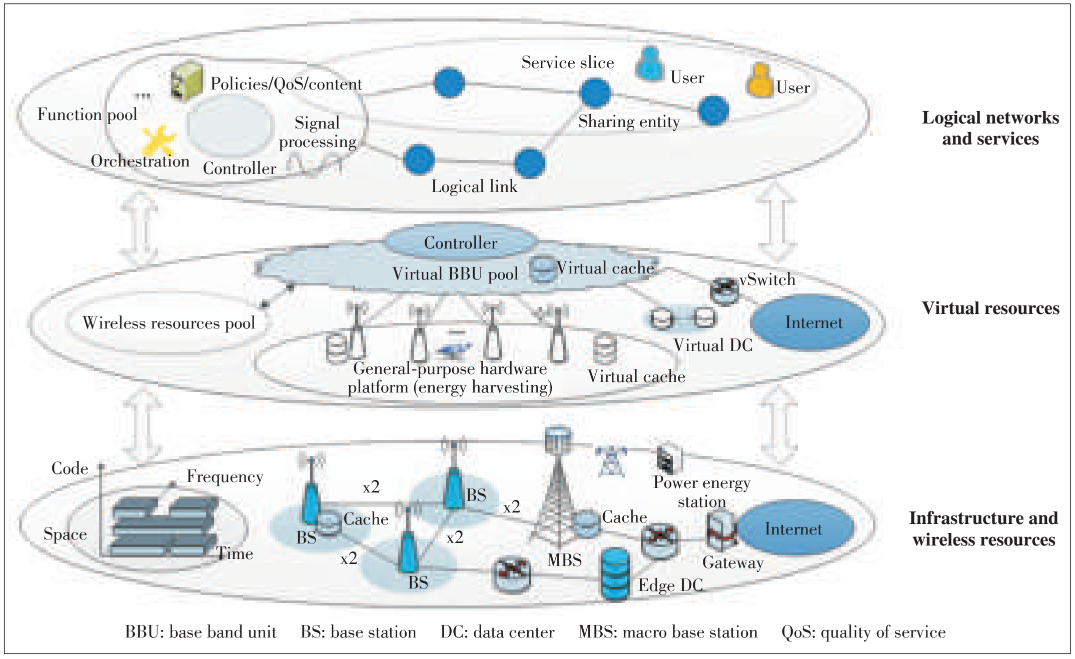


Figure 1. Network virtualization architecture from the resource perspective.

dense small cell via infrastructure re-engineering and sharing.

On the other hand, infrastructure virtualization also includes link virtualization as well as control and management. In [41], the processes of node virtualization and link virtualization, as well as management, have been concretely described. Specially, virtual links can be created by configuring Ethernet VLANs between the physical nodes hosting the virtual nodes. Actually, infrastructure virtualization may be directly observed in CC-RAN architecture in [42] and [43]; base band units (BBUs) are centralized and shared among different sites via virtualization, after which they are named a virtual BBU pool.

Since virtualization enables operation and management discrete, siloed infrastructure components toward a pooled infrastructure can be managed holistically. Adopting virtualized infrastructure can benefit a lot. For example, it can significantly reduce the complexity of deployment, and simplify operation and maintenance. Infrastructure virtualization can also create an elastic environment which helps business react more quickly to market demand changes and customization [24].

Fig. 2 gives an example for C-RAN architecture that applies the NFV architectural principle to an E-UTRAN based system. The enhanced NodeB (eNodeB) as a logical network entity is implemented both in the cloud platform and at physical radio access points (RAPs). The eNodeB consists of a number of virtual RAN processing units (vRPUs), responsible for executing the RAN protocol stack, and a virtual eNodeB controller that terminates 3GPP interfaces toward the core network and other eNodeBs. In the architecture, the virtualized eNodeB can be implemented as a virtualization network function (VNF), which is instantiated on a virtualized infrastructure.

Infrastructure virtualization can also achieve dynamical infrastructure resource allocation and traffic balancing. In this way, footprints and energy consumption can be lowered [44].

3.2 Radio Spectrum Virtualization

Radio spectrum resources are always scarce and significant for wireless communications. Generally, radio spectrum resources refer to the licensed spectrum or dedicated free spectrum [10]. With the emerging and developing of cognitive radio (CR) and edge computing technologies, the idle white spectrum can be used by others to alleviate the shortage, and relatively distributed computing for spectrum sensing has been applied for wireless networks. A novel wireless distributed com-

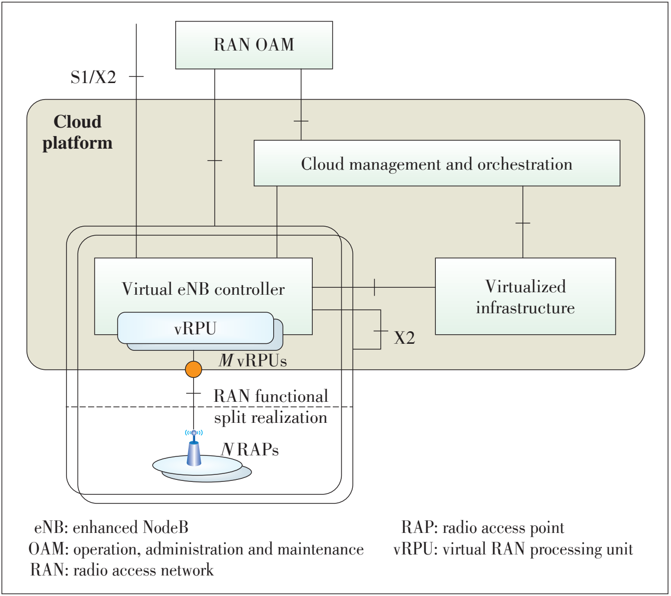

RAN OAM
S1/X2
Cloud platform
Cloud management and orchestration
Virtual eNB controller
vRPU
Virtualized infrastructure
X2
M vRPUs
RAN functional split realization
N RAPs
eNB: enhanced NodeB
OAM: operation, administration and maintenance
RAN: radio access network
RAP: radio access point
vRPU: virtual RAN processing unit

Figure 2. An example of infrastructure virtualization on C-RAN architecture [31].

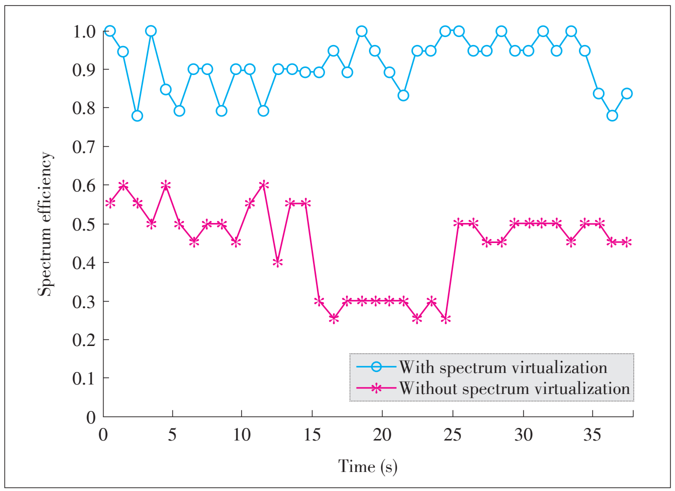
puting (WDC) scheme for cyclostationary feature detection spectrum sensing approach was proposed and investigated in [45]. Comparing with conventional Fast Fourier Transformation (FFT) time smoothing algorithms, the proposed scheme can reduce the computational complexity for each processing CR.

Spectrum virtualization considers the total available radio spectrum as a whole resource and virtualizes them as the abstracted access medium [22]. In [15], spectrum virtualization was viewed as an extension of dynamic access or sharing, which could achieve more spectrum sharing gain and improve wireless system capacity. A virtualization method and its procedures were also presented in [15], in which an advanced spectrum management (ASM) function obtains and updates related spectrum information, including available spectrum information from the cognitive plane and the latest spectrum usage information from eNodeBs, and the ASM function then allocates spectrum resource and coordinates among different eNodeBs. The evaluation results showed that after spectrum virtualization, the total utilization could increase by 30 percent (**Fig. 3**). In [18], a spectrum virtualization layer (SVL) under the physical layer was proposed to execute wireless spectrum virtualization. By spectrum reshaping techniques, the SVL could share the same RF front units on different ranges of the spectrum.

Spectrum virtualization can significantly promote full network virtualization, thus allowing extended radio spectrum shared by multiple operators [10] and higher spectrum efficiency (SE) and system capacity.

3.3 Cache/Storage Virtualization

Caching the multimedia contents at the network edge has been treated as one effective solution to dealing with the explosive growth of mobile multimedia traffic, improving the QoS of real-time data services, especially in strict-latency scenarios, and alleviating the heavy traffic burden on the backhaul and fronthaul. However, the effective in-network caching in traditional network architecture is not practical because of the dedi-



▲**Figure 3. Evaluation results of the spectrum virtualization in [15].**

cated hardware for signal processing in RANs and complex controlling and processing units in CNs, so the concept of "Cache-as-a-Service (CaaS)", a caching virtualization framework for cloud-based mobile networks was proposed and an illustration of the deployment of caching virtual machines was given as the Fig. 1 in [46]. CaaS is based on mobile cloud computing and a centralized caching controller is used to realize the caching VMs anywhere with properly allocated positions; by caching virtualization, the contents can be chunked, distributed, and stored based on its popularity, traffic diversity and the diversity of user requests.

Meanwhile, the content-level slicing of virtual resources (**Fig. 4**) was proposed in [29], which can be considered as an extension of dynamic content access and sharing through time multiplexing, space multiplexing, etc. There are three physical contents (caches) and three services sharing the contents, and physical cache is sliced into several virtual contents used by a service without knowing the existing of other slices.

3.4 Energy Virtualization

Energy efficiency (EE) has become a key pillar in the design of communication networks [47]. EE requirement mainly lies in two aspects: the 1000x improvement comparing to 4G networks on the network side and on the device side, especially for machine-type devices requiring a very long battery life (e.g. more than 10 years) [48]. The concept of Energy-as-a-Service (EaaS) has been proposed for reducing energy costs for both users and network operators [49].

On the one hand, mobile users can offload energy-consuming applications to cloud servers for limitations of battery capacity and computing resources. The related research can be found in [50]. This is a tradeoff between energy consumption and traffic latency for mobile users. On the other hand, since electromagnetic wave is almost ubiquitous, simultaneous wireless information and power transfer (SWIPT) may be one promising way to achieve the "recycling" of transmit power when receiving data services; in order to minimize the energy cost of data transmission in the context of collaborative mobile cloud (CMC) with SWIPT, the resources allocation and user scheduling were studied in [51]. In [52], a framework named GreenDelivery was proposed, where energy harvesting (EH) based small cells (SCs) provide content delivery services with respect to the battery status and content popularity distribution. The case studies show that related power consumption is reduced. If the EH based SCs are deployed, traditional passive energy consumers may be shifted to active energy prosumers.

The energy resource virtualization mainly focuses on how to achieve energy efficiency, regardless of offloading or SWIPT, EH or any other advanced schemes, such as traffic-aware service provision and wireless multicasting.

3.5 Data Center virtualization

Data centers (DCs) are facilities that house computer sys-

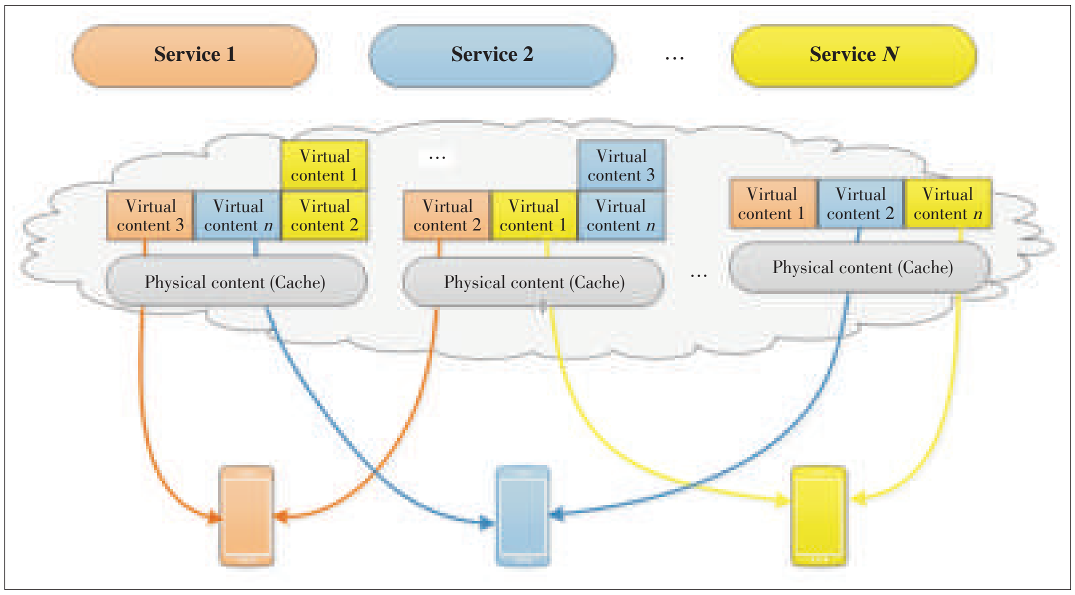


Figure 4.
An example of cache virtualization framework. The physical content (cache) is sliced into virtual contents that can be shared among different services dynamically and the slicing can be time multiplexing or space multiplexing [29].

tems and associated components such as networking and storage systems, and they are essential to satisfy ever-evolving computational demands around cloud computing, big data and IT infrastructure [53]. On the one hand, due to the delivery of explosively growing amount of data traffic over the wireless networks, considerable amount of wireless resources and costs will be occupied. Furthermore, it is expensive to build a lot of new DCs, so the best way is to improve usage of existing facilities with lower infrastructure overhead to achieve better resource management and cost efficiency [49]. On the other hand, data centers are facing some challenges such as the unguaranteed QoS, security risks, and management complexity and inflexibility, and the data center virtualization as an approach to address these challenges and virtualized data center embedding problem was studied in [54].

Data center virtualization is the process of designing, developing and deploying a data center through virtualization and cloud computing technologies, which encompasses a broad range of tools, technologies and processes that enable a data center to operate and provide services on top of virtualization layer/technology [55]. Using data center virtualization, an existing or a standard data center facility can provide/host multiple virtualized data centers on the same physical infrastructure, which can simultaneously be used by separate applications and/or organizations. This not only helps in optimal IT infrastructure/resource utilization, but also in reducing data center capital and operational costs. In addition, to reduce energy consumption, the approach of VM migration with server management and virtualization to increase the number of machines and switches that can be shutdown has been discussed [56].

4 Virtualization Architecture in CC-RANs

The RAN architecture has not experienced many changes since its birth in cellular system from the first generation (1G) to today's fourth generation (4G). Bases stations (BSs) are generally deployed and operated in a distributed way with the hardware and software tightly coupled [57]. Under the traditional RAN architecture, since control/user (C/U) planes are coupled, some BSs cannot fall asleep or be switched off even though there is little traffic because of basic coverage, leaving some extra energy wasted. Therefore, capacity and coverage need to be separated via logically decoupling the data delivery and control signaling distribution in future R-RAN systems [58]. The network deployment and upgrade are also expensive and time consuming due to dedicated hardware and distributed deployment. In addition, facing the exponentially growing mobile Internet and IoT traffic as well as pursing 1000 times EE improvement, the RAN architecture and corresponding deployment are expected to achieve some breakthroughs.

On the one hand, cloud based network architecture takes the advantage of the centralized cloud principle to share storage and computational resources via virtualization [42], which can achieve powerful processing and dynamical resource allocation. It is worth to note that the C-RAN is a NFV instance on the RAN side to achieve soft RAN [59]. Actually, virtualization technology has been applied in CC-RANs. As mentioned in the Introduction section, the CC-RANs include the C-RAN, the H-CRAN, and even the F-RAN. All of them are integrated with virtualization technology but have some inherent bottlenecks or challenges.

On the other hand, the development of SDN is tightly connected to cloud computing, since cloud computing makes large-scale logical centralized control solutions feasible, including centralized data storing, processing, online accessing, etc. [60]. In addition, some researchers have introduced SDN into fronthaul networks named as SDF network in C-RANs [57], [61]. Therefore, the software-defined characteristic has a tight relationship with the cloud computing, and we will give a brief review on virtualization architecture in SD-RANs, C-RANs, and F-RANs. We can discover that it is a trend that the virtualization architecture in F-RANs will be integrated with SDN, network virtualization and Cloud computing. Here, the cloud computing includes cloud computing and fog computing, since the fog computing is usually treated as an extension of classical cloud computing.

4.1 Virtualization Architecture in SD-RANs

SDN has brought a rethinking of routing and packet switching in the Internet [57]. Introducing the principles such as the C/U split, centralized control and software application programming interfaces (APIs) into RANs has been proposed in academia and industry. In [61], self-organizing networking (SON) solutions and the concept of SDF as the RAN optimization using above SDN principle were discussed. In [33], in order to enable multi-RATs interwork and heterogeneous deployment, one software-defined access (SDA) architecture was proposed and discussed. In [62], in order to achieve the conception of efficient spectrum management and sharing mechanisms, a software-defined 5G heterogeneous network (HetNet) system was introduced to support harmonized SDN-enabled approach.

In the SD-RANs, macro-cell BSs are directly connected to the controller, while the BSs of smaller cells connected with the macro-cells via a reliable backhaul link. In [63], SoftRAN was proposed as a software-defined RAN. By abstracting all base stations in a locally geographical area as a virtual big base station composed of a centralized controller and individual physical base stations just with minimal control logic, the SoftRAN was introduced as a software defined centralized control plane, which has refactored the control plane functionalities. The latency-sensitive decisions continue to be handled by the individual physical base station. In SoftRAN architecture, the global utility over a logical geographic area can be optimized and network management can be simplified by software programming.

In [58], based on the SD-RAN, a beyond cellular green generation (BCG2) architecture base on LTE system was proposed, which decouples the centralized control plane and data-forwarding, as well as decoupling coverage and capacity. The eNodeB functionalities are split between the signaling nodes and data BSs. If there is no demand, the data BSs are in asleep mode, while the signaling controller is always-on for coverage. Since the signaling node is designed based on general-purpose processors and can be sharing by operators via virtualization, the power consumption will not substantially increase. In [64], the SD-RAN controller for enterprise WLANs and the implementing platform were introduced, in which some of the main features of the SD-RAN controller were discussed, including slicing (a network slice is a virtual network), soft state, and modular architecture. **Fig. 5** shows the virtualization architecture in SD-RANs.

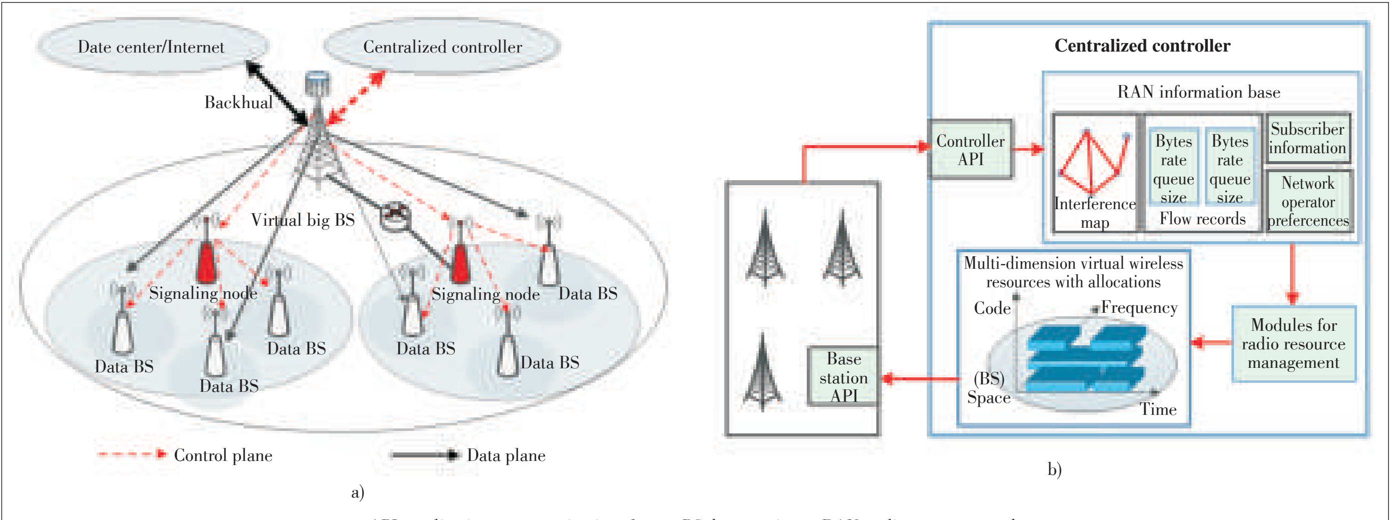
In Fig. 5a, the individual physical base stations include signaling nodes and data BSs. Signaling nodes are only responsible for control plane functionalities, while data BSs mainly focus on data transmissions and some delay-sensitive decisions with minimal control logic. However, through software-defined way, signal nodes and data BSs can be shifted to the others. Fig. 5b shows the centralized controller, referencing to [58], where the virtual big BS communicates with centralized controller via related APIs. It is worthy to note that the centralized controller has a global view of the SD-RAN. In the state-of-the-art proposals of SD-RANs, the centralized controller resides in either the core network or a centralized data center.

However, since the limited processing ability and flexibility of the centralized controller, the individual "virtual big BS" in the SD-RANs may fail to do well in large-scale cooperation processing (LSCP) and massive connection scenarios. It may also be caught in a dilemma in the latency-sensitive missions because of the simple control logic of data BSs and the delay between users and centralized controller. Through the signal nodes realizing control plane functionalities in a locally geographical area, the hierarchical design may perform well enough in the sparse deployment, but it is unable to effectively handle rapidly growing traffic and the dense base station deployment [63]. In addition, the virtualization gain of SD-RANs mainly comes from wireless resource virtualization and the centralized controller behind virtual big BS, leaving the CAPEX and OPEX saving not being considerable.

4.2 Virtualization Architecture in C-RANs

The SD-RANs are facing above challenges and urgent breakthroughs need to be made in RAN architecture for future network requirements. On the other hand, as a promising solution to reduce CAPEX and OPEX, the C-RAN architecture can provide high SE and EE [42]. The C-RAN architecture is a significant representative of cloud-based RANs, since underlay heterogeneous network (HetNet) with cloud computing (HetNet-CC) [65] and H-CRAN [66], [67] are based on cloud computing [68]. It offers novel schemes and architecture for centralized management and flexible network resource allocation.

The C-RAN architecture is the typical CC-RAN architecture with virtualization. Based on cloud computing, the traditional BSs are decoupled into two parts: the base band units (BBUs) migrated into a BBU pool and the distributed remote radio heads (RRHs) that are connected to the BBU pool via fronthaul links [69]. The authors of [42] have comprehensively surveyed the C-RAN architecture, including the general architecture, the systems proposed by industries and academia, and the architecture toward 5G network, so we omitted the C-RAN architecture here for brevity. Most functions of traditional BSs are virtualized as a centralized BBU pool [70], upon which multi-MNOs can share the physical infrastructures and VMNOs can share the virtual network resources according to the demands of SPs and users. Meanwhile, MNOs can quickly deploy RRHs



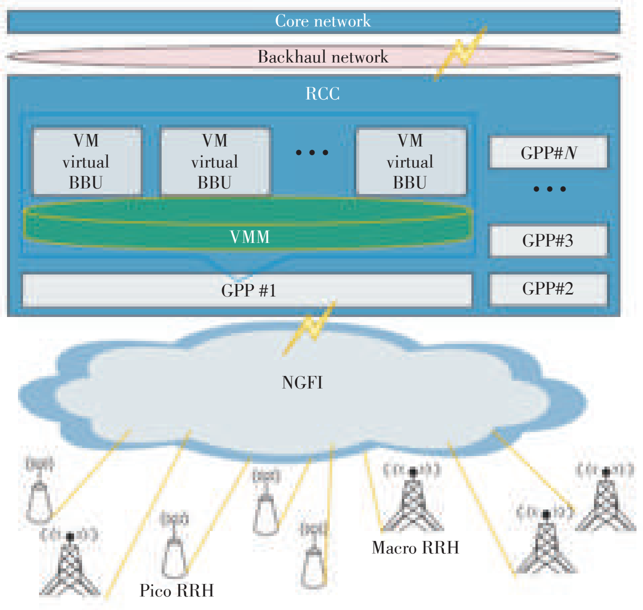
API: application programming interface BS: base station RAN: radio access network

▲Figure 5. Virtualization architecture of a) SD-RAN and b) the centralized controller [63].

to expand and make upgrades to their cellular networks [42]. MNOs and VMNOs can also scale the computing resources of the cloud up and down depending on the live traffic demands, which can further improve the network resource utilization. However, a very important aspect to lower costs depends on how to realize the network functions of BBU pool in cloud [71]. Through virtualization technology, the BBU pool can be shared by different sites and among multiple MNOs. In [71], a BBU pool was built in the cloud based on SDN and network function virtualization concepts. In the pool, the hardware resources are virtualized and allocated to service manager by infrastructure manager on demand, which depends on the requirement of a BBU virtual instance. In [59], a C-RAN network architecture combing the ideas of SDN and NFV was introduced, and the idea that C-RAN is an NFV instance on the RAN side to realize soft RAN was illustrated. We treat the architecture as a typical virtualization architecture in C-RANs.

In **Fig. 6**, the C-RAN architecture consists of a radio cloud center (RCC), next-generation fronthaul interface (NGFI)-based fronthaul (FH) network, and RRHs. The RRC provides a cloud platform with all kinds of sharing resources and all the CC-RAN functions appear as software applications running in VMs under the management of one hypervisor/virtual machine monitor (VMM) [72]. The RCC may also provide isolated general-purpose platform (GPP) to network operators to develop new services or carry out an experiment without influencing existing operations. The baseband-related functions are processed by the BBU while the radio frequency related functions are processed by RRHs; therefore, the traditional common public radio interface (CPRI) is more and more unsuitable for future networks with networks evolving to 5G, because of its constant data rate, relatively fixed connection between the RRH and the BBU, and the sampling I/Q data rate dependent on antenna amount [59]. The next-generation FH interface has been designed. Rethinking the split BBU-RRH function, there will be several kinds of RRHs. One new RRH may contain RF functions and partial baseband processing functions. Especially, the authors have mentioned one RRH may be able to be automatically switched to another BBU pool for protecting communication reliability.

Some researchers have also introduced SDN into SDF network in C-RANs [57], [61], and the development of SDN is tightly connected to cloud computing, since cloud computing makes large-scale logical centralized control solutions feasible



BBU: base band unit
GPP: general-purpose platform
NGFI: next generation fronthaul interface
RCC: radio cloud center
RRH: remote radio header
VM: virtual machine
VMM: hypervisor/virtual machine monitor

▲Figure 6. Virtualization architecture in C-RANs [59].

[60]. Recent literature has started rethinking the C-RAN architecture with combining multiple virtualization technologies. Based on the idea of SDN, including SD-RAN [63], SDF [58], and the NFV on cloud infrastructure, the software-defined hyper-cellular architecture (SDHCA) has been proposed in [57] as its Fig. 1.

In the physical representation, the architecture can be divided into three subsystems. It is worthy to note that the RRHs have flexible functions. They can merely execute RF transmission/reception, or be dynamically configured to play the roles as control base stations (CBSs) and traffic base stations (TBSs) with some baseband processing functions. Especially, the deployment of RRHs can be compatible with the traditional network planning mechanisms, thus realizing the largely multiplexing of traditional BSs. On the other hand, the proposed SDHCA can provide one control coverage and multiple conceptual layers for different user traffic types as we can see in the logical presentation.

Concretely, through software-defined and function virtualization on the cloud infrastructure, the RRHs can be flexibly configured and the network functions can be split from hardware. The control and data planes of SDF network are also decoupled, which may be one direction to deal with the constraints of fronthaul capacity and coat-inefficiency [69]. By software applications running on VMs, the network functions, such as air interface control, service analysis, and fronthaul control, can be easily programmed and updated. In addition, the computing resource utilization may be higher since customization and the power consumption may be decreased in the sleeping model.

4.3 Virtualization Architecture in F-RANs

The traditional C-RAN architecture [42], [73], [74], with centralized processing architecture and fronthaul bandwidth constraints, may lead to time latency, unreliability, and jitter, especially when traffic load is heavy [69]. At the same time, since the emerging wave of the IoT and latency-sensitive applications, a new platform called fog computing has been proposed in [75] to satisfy such requirements as location awareness, mobility support, distributed deployment and low latency. Fog computing extends the cloud computing paradigm to the edge of networks, while fog and cloud use the same resources (networking, computing, and storage), and share many of the same mechanisms and attributes, such as virtualization and multi-tenancy [76]. Fog computing, cloudlet and mobile edge computing as three typical edge computing technologies were introduced and compared in [77], and the network slicing in edge computing was pointed out as a challenge, which may be realized via RAN slicing based on virtualization technology.

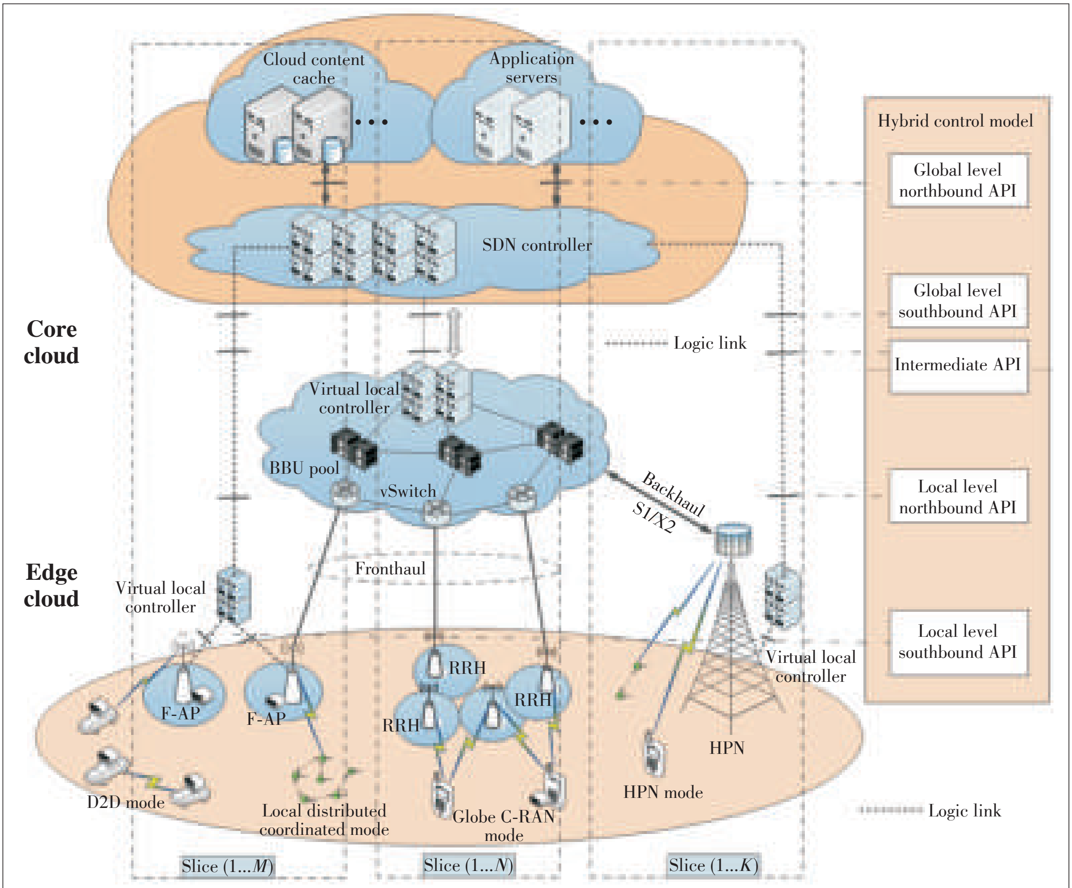
In [78], an integrated architecture for software-defined and virtualized RANs with fog computing was proposed to deal with IoT scenarios, since the tremendous number of communication devices connecting to wireless networks imposes huge challenges on network deployment, management, and data processing. The core idea of the architecture is that SDN decouples the control plane and data plane to provide network programmability, and virtualization realizes the sharing of network and radio resources among virtual slices, as well as the benefits of fog computing at the edge of networks. In addition, a virtual resource chain (or network - level virtualization) called Openpipe, and interface architecture was proposed. The Openpipe can be treated as an end-to-end network slice. Similarly, the SDN and network virtualization can be applied into F-RANs, which will bring great benefits into F-RANs.

The F-RAN has been presented in [79] as a promising paradigm to provide high SE and EE, while maintaining low latency for future wireless networks. Through local radio signal processing, cooperative resource management, and distributed storing capabilities in edge devices, F-RANs can decrease the heavy burden on fronthaul and avoid the large-scale radio signal processing in the centralized BBU pool, both of which can decline delay and improve resource utilization.

The related research on F-RANs has been widely done. In [79] and [80], the system architecture, key techniques such as transmission mode selection and interference suppression of F-RANs, and performance analysis based on cache, mode selection, radio resource allocation were studied for optimizing SE, EE and latency. In [81], combing the advantages of C-RAN and F-RAN, the F-CRAN was proposed as the harmonization architecture. In [82], an information-theoretic framework was introduced to capture the key tradeoff between delivery latency and the capacity of fronthaul and caching storage, with a conclusion that edge caching, fronthaul and wireless transmission should be jointly designed to leverage synergistic and complementary advantages of edge processing and virtualization.

However, as discussed in Section 1, F-RANs are also facing the challenges such as huge amount of power consumption and carbon dioxide CO2 gas [83], the complex management and mobility support requirements, and compatibility for multi-RATs due to its heterogeneity and distribution characteristics. Moreover, the cost efficiency, flexibility and high resource utilization are always the pursuing of network deployment. In addition, differentiated IoT application scenarios, such as connected vehicle, smart grid, wireless sensors and actuators networks, [75] need customized network performance, such as low latency and massive connectivity. The F-RAN virtualization may be a powerful way to deal with the above challenges.

Based on the fog computing architecture in [76], and the F-RAN architecture in [79] and [80], we design the virtualization architecture in F-RANs (**Fig. 7**). The architecture consists of the core cloud and edge cloud. In the core cloud, the centralized BBU pool is designed with virtualization technologies, such as NFV [25]. In the edge cloud, there are fog access points (F-APs), each of which integrates the front RF, the local distributed collaboration radio signal processing (CRSP), the cooperative radio resource management (CRRM) and caching capabilities, RRHs, UEs and other communication nodes in di-



API: application programming interface
BBU: base band unit
C-RAN: cloud radio access network
D2D: device-to-device
F-AP: fog access point
HPN: high power node
RRH: remote radio header
SDN: software-defined networking

Figure 7. Virtualization architecture in F-RANs.

verse service scenarios. All these functions have their performance requirements, such as high rates for mobile Internet services, ultra-low delay and high reliability in vehicle networks, massive connections in intelligent metering and measurement with adaptive transmission mode selection. Based on virtualization technologies and network slicing, there are multiple slices coexisting on the same physical network.

In order to better deal with the distributed and hierarchical characteristics in F-RANs, the architecture in [84] adopted the hybrid control model of SDN controller as Kandoo. The model SDN controller (in the global level) defines the specific policies and sends them to local controllers (in the local level, which can be in a slice) through intermediate interface (intermediate API).

5 Key Enabling Technologies in Virtualized CC-RANs

The enabling technologies for wireless network virtualization have been presented according to different RATs in [22]. The isolation level, control method, and purpose were also mentioned to show that these technologies can be used as a taxonomy to classify key technologies. We have reviewed some related literature on virtualization in the context of RANs, such as [19]–[21] and [31].The requirements for RAN virtualization have been universally mentioned as customization, isolation, and high resource utilization. And since the RAN architecture with virtualization technology, such as sharing GPPs and other virtual resources, the C/U split and software-defined has broken the traditional networking and resource allocation ways, and new challenges and corresponding key enabling technologies are on the way to RAN virtualization.

5.1 Requirements of RAN Virtualization

In this section, the state-of-the-art key enabling technologies for virtualization in CC-RANs are presented as virtual resource allocation, RAN slicing, mobility management, and social-awareness, all of which are mainly encompassing on how to realize RAN virtualization properties and requirements, namely customization, isolation, and high-efficiency utilization of radio resources.

5.1.1 Customization

Customization mainly includes the resource allocation customization and the flexible architecture reconstruction to satisfy related network service demands. In [20], customization stresses that network NVS should provide simple and appropriate programming interfaces to enable customized solutions within slices (a slice consists of all the flows of an entity that requests virtualized resources), which is resource allocation customization. Similarly, in [19], through providing flexibility to different entities, these entities can program the vBS to optimize their service delivery. One resources negotiation for network virtualization (RENRV) algorithm has been proposed in [21], where resource customization is attained during the detection phase, and the requesting BS will be allocated resources according to its actual traffic load conditions. As to the architecture aspect, obviously, since the traditional "one-size-fits-all" network solution today is unable to satisfy future network service demands well, the Nokia analyzed dynamic end-to-end network slicing, which is introduced to establish multiple parallel network instances and realize customization in architecture reconstruction [85].

5.1.2 Isolation

Since multiple virtual network slices should coexist, isolation is the basic issue in virtualization. Isolation means that any changes in one virtual slice, such as the number of end users, mobility of end users and fluctuating of channel status should not cause any serious interference and negative influence on resource allocation for other slices [20]. In [86], isolation among slices was introduced as the essentiality about managing network and computing resources to realize a picture in which the performance of one slice is not affected by the operation of another slice. Moreover, the protection mechanisms were mainly concentrated on protecting common (control) channels or resources used for UEs accessing system.

However, each slice may be assigned with either shared or dedicated radio resources up to radio resource management (RRM) implementation and service level agreement (SLA) [87]. Therefore, different use cases and service requirements will need different isolation levels, which refers to the minimal resource granularity [22], in which the isolation ways, such as MAC layer, packet, flow, multilevel, and even traffic types and context, have been surveyed from sub-carriers, sub-channel, PRB, time-slot, space, upper layers and protocols. However, the ideas of isolation is mainly using orthogonal resources. In [21], the isolation is achieved during the transfer phase, through Resource nEgotiation for Network Virtualization (RENEV) algorithm ensuring a reserved portion of resources to each requesting BS, the isolation and resource demand can be satisfied. The two above methods may have low resource utilization with low complexity.

Actually, using completely orthogonal resources to realize isolation may be not practical, since the resources are limited, especially in RAN virtualization with scarce spectrum resources. Therefore, the interference or negative effects under one threshold may be able to treated as an isolation. From the discussion above, the isolation can be considered from the protocol design such as MAC scheduling protocols, and resource allocation.

5.1.3 Resource Utilization

Given the RAN virtualization is treated as the RAN resources mapping to multiple virtual RAN slices, resource utilization can be defined as the actually mapped resources of one slice divided by the total amount of available resources for all virtual slices [22], [88]. To evaluate the resource utilization of a virtual slice, it can be derived with the used virtualized infrastructure resources, bandwidth, power, time-slots, and processing capacity resources divided by the relatively available resources [22]. In [19] and [20], dynamically adjusting resources across virtual slices to maximize the resources occupied as much as possible was discussed. In [21], the RENEV guarantees the efficient physical radio resource utilization by reconfiguring the medium access of each pair of requesting-requested BSs with a rational signaling burden.

Finally, RAN virtualization should guarantee the efficient use of physical radio resources, computing resources, and other resources. The virtual resource allocation will play a significant role in higher resource utilization, which we will discussed in next subsection. At the same time, the virtualization technology needs to cope with the tradeoff between complexity and efficiency in dynamic resource allocation.

5.2 Virtual Resources Allocation in CC-RANs

The problem of embedding virtual networks in a substrate network is mainly the resource allocation in network virtualization [88]. Similarly, in CC-RANs, the virtualization can largely be treated as the allocation of virtualized infrastructure resources and wireless resources, thus the virtual resource allocation plays a vital role in the CC-RAN virtualization. The final goal of virtual resource allocation is to improve the resource utilization as much as possible with pursuing the higher system capacity and satisfying SLA requirements. However, the following has to be taken into consideration for different service scenarios: diverse service requirements, traffic fluctuation characteristics and the inherent characteristics of wireless environment such as broadcasting and stochastic fluctuation of channel quality, different resource granularity and resource priority, and the heterogeneity of IoT devices. For example, ultra-low latency services need more computing resources while high data rate services should be provided enough spectrum resources. Therefore, in virtualized RAN environment, among multiple virtual CC-RAN slices, the virtual resource allocation will be more challengeable. Fortunately, there have been some related works on virtual resource allocation.

Considering the fog computing at the edge of network and the increment of resource dimension (radio resources, edge caching and edge computing resources, and virtual network functionalities), the virtual resource allocation in virtualized F-RANs can be carefully designed from two significant aspects.

One is virtual resource allocation framework, and the other is joint resource allocation and resource granularity.

Virtual resource allocation has been widely discussed mainly based on a hierarchical architecture, which is suitable for virtualized F-RANs. since the diverse IoT devices and service scenarios as well as the distributed characters of F-RANs, the hierarchical architecture can be more efficient and flexible.

In [89], a hierarchical virtual resource allocation model was proposed to maximize the MVNO profit with the consideration of backhaul capacity of the infrastructure providers (InPs) and users' QoS requirements. In [90], a base station equipped with a large number of antennas serves users who belong to different virtual slices (a typical hierarchical scenario) was analyzed and joint power, sub-carrier and antenna allocation problems were studied to maximize the sum-utility while maintaining the minimum rate for each slice. In [91], a joint BS assignment, sub-carrier, and power allocation algorithm for dynamic resource allocation was proposed to maximize the network sum rates under the constraint of maintaining the minimum required rate of each slice (the users of different service providers) in multi-cell virtualized wireless networks.

Relatively, the models or methods to solve virtual resource allocation problems become more interesting. For example, the successive convex approximation (SCA) and complementary geometric programming(CGP) were adapted to develop an efficient two-step iteration approach to settle above virtual resource allocation problem in [91]. Since all involved parties, including InPs, MNOs, MVNOs, and SPs, want to maximize their own revenue, the game theoretic approach may be an effective tool [6], [92]. In [6], in order to address two-level hierarchical virtual resource allocation problem for achieving strict inter-slice isolation and the ability of intra-slice customization, a hierarchical combinatorial auction mechanism was mentioned. In [92], the dynamic interactions among the SPs and the network operators (NOs) were modeled as a stochastic game, where SPs are responsible for QoS management for their own users while NO manages the spectrum resources and regulates the game.

We can find the pervasive feature in virtual resource allocation in RANs is based on a hierarchical architecture with the introduction of MVNOs and slices for differentiated services and flexible management. The other outstanding characteristic is resource dimension increment in virtualized F-RANs. Because of the edge caching and edge computing ability of F-RANs as well as infrastructure and network function virtualization, traditional radio resource allocation may not be able to satisfy QoS requirements. The deployment problems of partial core network functionalities based on VMs and virtualized links and nodes mapping also need to be dealt with in virtual resource allocation.

In [13], virtual network embedding (VNE) and NFV resource allocation (NFV-RA) problems were discussed. VNE deals with the allocation of virtual resources in nodes and links mapping to substrate nodes and links, respectively. The VNE problem is known to be NP-hard [93] and most of the related work has been concentrated on designing heuristic algorithms and simplifying the network realization complexity. Meanwhile, VNE can be optimized by embedding cost, link bandwidth, QoS, and energy efficiency. On the other hand, NFV-RA has three different stages, including VNFs-chain composition (VNFs-CC), VNF forwarding graph embedding (VNF-FGE), and VNFs scheduling (VNFs-SCH). The VNFs-CC seeks to concatenate the VNFs efficiently for a network service in the most adequate way, and VNF-FGE seeks to find where the VNFs will be allocated in the NFVI in a suitable way, considering the requirements of individual requests and the overall requirements of all services. The VNFs-SCH seeks to determine when is better to execute each function into the NFVI to minimize the total execution time without degrading the service performance and respecting all the priorities and dependencies between the VNFs composing the service at the same time. In [8], the scheduling of wireless VNFs in the RAN was formulated as an integer programming problem. In [94], a management and orchestration (MANO) entity in virtualized infrastructure has been enabled. In addition to deriving the affinity scores for resources units (RUs) referencing to a specified RU for concrete VM instance, MANO can perform precise and efficient resource tailoring and dimensioning, which benefits the operation and management of the virtual network functions in virtualized infrastructure. Both [13] and [94] focus on virtualized infrastructure resource allocation that can be applied in that of F-RAN virtualization.

The joint resource allocation on virtualized CC-RANs also plays a vital role, since network performances such as service latency, energy efficiency and spectrum efficiency are influenced by multiple factors. When the resources are virtualized, the extended sharing and flexible allocation should be more elaborated. The joint optimization of cloud and edge processing, focusing on the strategy to populate the caches of enhanced RRHs (eRRHs) for F-RANs, can be found in [95].

Comparing with traditional static resource allocation, virtual resource allocation is expected to gain better performance, and the model and methods of resource allocation and optimization need carefully redesign with considering the resource granularity and dynamically sharing characteristic. In addition, virtual resource allocation should be further researched, especially for real-time and low-complexity demands since the virtual resource mapping processes and generic hardware with low cost may decrease specific performance.

5.3 RAN Slicing

The current relatively monolithic network architecture is not flexible and scalable enough to efficiently support future networks with diverse and sometimes extreme service requirements. In addition, the introduction of network services should be efficient and multiple use cases are expected to be operated on the same operator network [96]. Network slicing has been regarded as one of the most significant technologies to deal with the above challenges.

In [38], the concept of network slicing as a service (NSaaS) was introduced to offer end-to-end cellular networks as a service. Three forms of networking slicing, including CN only, RAN only, and CN and RAN were also introduced. The RAN only slicing runs on wireless platforms, which consists of radio hardware and baseband resource pool. Both radio hardware and baseband resource pool can be realized on GPPs, where various parameters of air interfaces, such as symbol length, sub-carrier spacing, cycle prefix length, and the parameters of hybrids automatic repeat request (HARQ), as well as other parameters like cell selection, handover threshold, can be set up for each slice to realize logical network customization [38].

Recently, RAN slicing has been designed mainly from two aspects. One is CC-RAN resource management perspective, and the another is RAN architecture and protocol stack perspective. Through the architecture and protocol stack design, diverse logic RANs can be achieved based on the unified physical network according to SLA requirements. The SLAs can describe which QoS metrics are expected to be guaranteed, possibly denoting a minimum amount of spectrum.

From the RAN resource management perspective, the authors of [97] presented RAN slicing at four levels, including spectrum planning level, inter-cell interference coordination (ICIC) level, packet scheduling (PS), and admission control (AC), and compared them from the granularity of isolation (i.e. high level of radio-electrical isolation and low level of traffic isolation) and customization. It was concluded that the RAN slicing approaches providing high level isolation may lose the higher granularity and flexibility. In [98], the network slicing architecture featuring CC-RAN abstraction was proposed, which uses the FlexRAN concept [99] to enforce network slicing in the RAN and adapt the resource allocation policy according to the slice requirements. From the level of resource isolation, the dedicated resources and shared resources models were aslo mentioned, as well as a two-level MAC scheduler to abstract and share the physical resources among the slices.

From the protocol stack perspective, in [86], multiple RAN slices may multiplex into a common MAC, PHY and radio, while other slices may use dedicated spectrum or dedicated APs, and thus use a dedicated MAC/PHY that involves new protocol design and model selection. In [100], Chain Mobile and other companies have designed the RAN architecture with network slicing, which emphasizes that the RAN protocol stack can be tailored to meet diverse service requirements of different network slicing instances (e.g. enhanced mobile broadband (eMBB) slices, ultra-reliable and low latency communications (URLLC) slices, and massive machine-type communications (mMTC) slices, etc.).

In addition, due to the particularities of wireless environments, guaranteeing slice isolation is full of challenges [101]. As for AN slicing isolation, similar to the level of resource isolation mentioned above, there are mainly two ideas. One is that RAN slices occupy completely orthogonal resources at a resource granularity interval. The other is maximizing the network resource utilization with controlling the negative effect or radio interference under a certain threshold. The former can be easily deployed but with the low resource utilization, which can be used for low service traffic load and needs no extra overhead for interference management. The latter will be complex and needs other extra assisted technology such as intelligent spectrum sensing, interference cancellation, and other advanced virtualization or software migration technologies.

5.4 Mobility Management

Mobility management mainly includes location management and handoff management. The former enables content delivery and communications, while the latter mainly keeps service continuity when users move from one access point to another [102]. Generally, users move freely and independently, requiring ongoing communications and minimal disruptions. This may leads to stochastic and time-varied interference to others. In real network deployments, user mobility and interference in a multi-cell and multi-layer network need to be fully considered, which has a direct influence on the user experience and system capacity.

With virtualization applied in CC-RANs, virtual resource allocation, extended radio spectrum sharing, and hierarchical management architecture will leave more challenges on mobility management. For example, one user's location update may involve multiple different VMNOs or InPs, which makes the tracking location complex.

Related research has been done. In [103], based on wireless network virtualization technology, a handoff scheme was introduced to support handoff between three virtual networks, each of which has a network controller used to schedule users, in an integrated train-ground communication system.

In [104], a SDN-enabled authentication handover and protection mechanism was designed for dense and heterogenous small cell networks for 5G, which is mainly based on sharing of user-specific security context information among related APs. An authentication handover module was also implemented in the SDN controller to monitor and predict the location of users. The relevant cells prepare related resources in advance, and the user can get seamless handover authentication during mobility.

Since SDN has been introduced into virtualized F-RANs, the local controls are deployed in the edge cloud, which can support high level of SDN controller to make global handover decisions and can handle handovers in F-RAN slices. In [105], one scheme of mobility management for network slicing based on 5G system was proposed. **Fig. 8** shows the handover procedure, which is based on the SDN controllers in edge cloud as well as a hierarchical and cooperative control mechanism. The mobility management can be employed in virtualized F-RANs.

5.5 Social-Awareness

Social-awareness is also known as social consciousness and originates from sociology, which is often used to describe the sociality and social behaviors of human beings [106]. Social-awareness has been introduced in communications, which is originated to enhance D2D communications for relieving energy consumption, data usage [107], cutting down transmission delay, etc. Recently, Social-aware energy harvesting D2D communications [108] and caching based socially-aware D2D communications [109] have appeared in literature.

Since rich diversity of mobile devices with powerful computation capacity, social-awareness has gradually been used for networking. By exploiting the users' social relationships and behaviors, the social properties of nodes can be analyzed and used. In [106], the socially aware networking (SAN) was surveyed. From the architecture of the SAN, we can see that the first two steps are to obtain social-awareness by sensing and analyzing data by using intelligent technologies, such as data mining and machine learning. Then, the SAN can deduce important social properties, including community, centrality, similarity, tie strength, and human mobility pattern. Based on these social properties, different networking protocols to satisfy the requirements of applications can be adaptively designed.

Social-awareness can be applied in F-RAN virtualization, especially in the cache/storage virtualization, energy virtualization, virtual resource allocation, and dynamically networking.

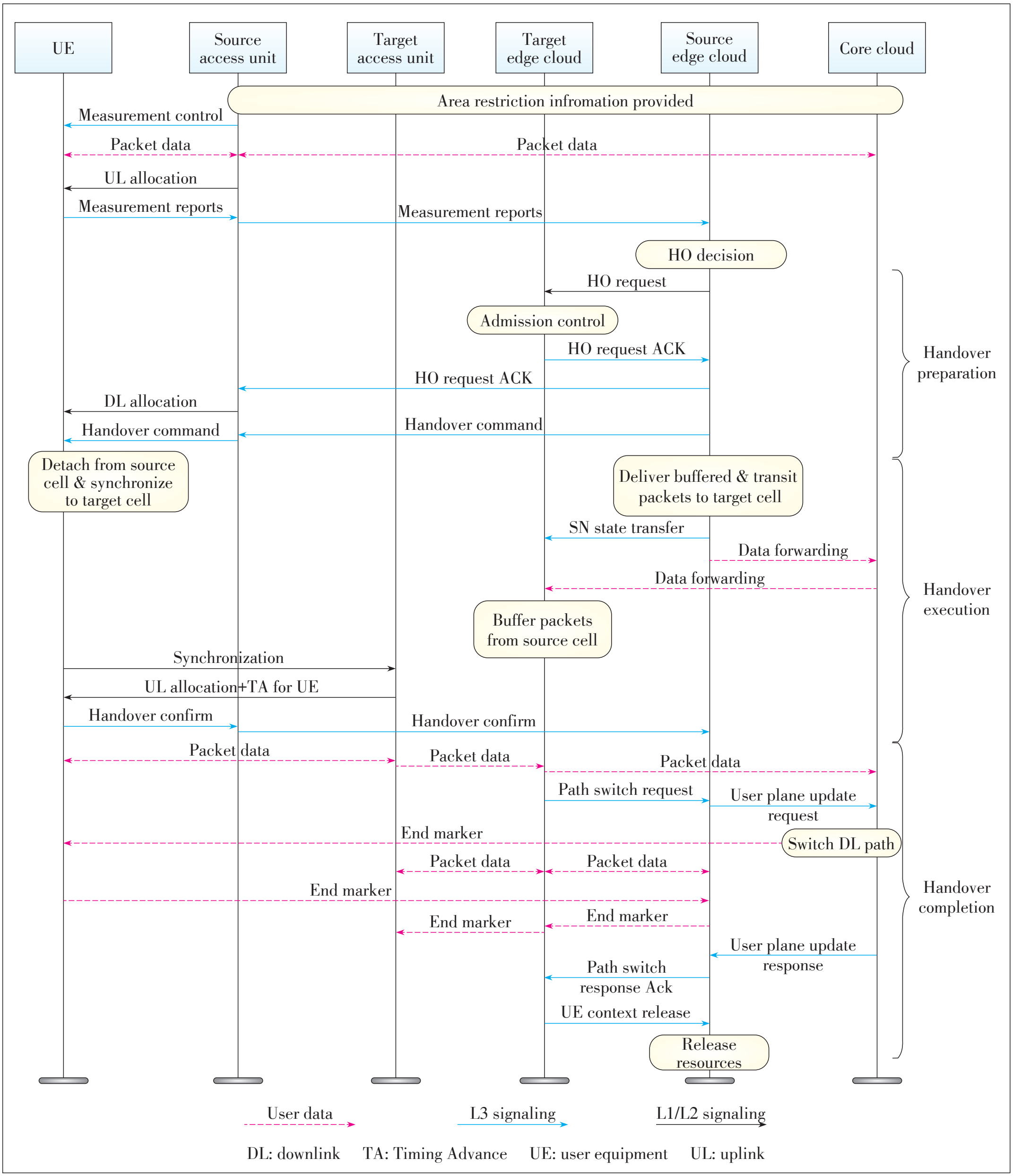


Figure 8.
The handover process for virtualized F-RANs based on 5G network slicing system [105].

In a virtual environment, the entire system, including the operating system, the applications, and data, can be encapsulated as a set of VM files [24]. Based on the social-awareness, the probable locations of VMsand network resources can be predicted. With the deployment and migration in advance, the time of service request response can be declined and the resource utilization can be improved. In [110], the social-awareness was introduced into virtual network embedding (VNE) paradigm, which mainly contributes to virtual resources efficiently mapping onto physical ones. Since social-awareness provides useful information such as the importance of a node based on specific mapping algorithms, the physical resource allocation among the virtual node requests can be more efficient since the popular nodes will be initially selected.

6 Challenges and Open Issues

Virtualization technology has been applied in IT area for many years. With the explosively growing traffic of diverse services and the demand of quick network deployment with cost-efficiency, virtualization technology has been introduced into communication networks. However, there are still many challenges in the virtualization of CC-RANs before its wide and successful rollout.

6.1 Virtualization Levels for CC-RANs

Resource sharing has been fully discussed at the spectrum level, infrastructure level, and network level in H-CRANs in [28]. In [101], slicing could be done at different levels, from application and flow slicing to hardware and spectrum slicing. Different levels mean different benefits and challenges. Similarly, the virtualization levels represent different sharing and abstraction degrees, which influences the resource granularity and the system flexibility and complexity. In real deployments, it is necessary to decide up to which level virtualization should be applied to achieve an efficient virtualization solution. In CC-RAN virtualization environment, virtualization levels mean different protocol stack design. How to determine the virtualization levels and describe them are worth of further investigation.

6.2 Signaling Design for CC-RAN Virtualization

The process of virtualization maps multiple virtual networks to the same physical network with the logically centralized control function. The controller or hypervisor takes responsibility of the signaling generation and forwarding [72]. In addition, in each virtual network (or a slice), some more elaborate control signalings directly satisfy users’ service requirements. Considering the differences among virtual networks, the differentiated signaling design or unified signaling set design should be analyzed. The unified design is easily generated, processed, and upgraded, while the differentiated design is more streamlined. The present common sense is that proper control signaling design needs considering delays and reliability and should be explored in a careful manner to enable the connectivity among different parties involved in CC-RAN virtualization [49].

6.3 Performance Analysis for CC-RAN Virtualization

With virtualization technology applied in wireless networks, the software-defined and functional modular design increases the difficulty in measuring the performances because of the flexibility and other constraints. As shown in [111], the initial virtualized network function (VNF)/VNF components (VNFCs) deployment strategy needs to satisfy the intra-functional constrains between multiple VNFCs as well as administrative constraints; the VNFC embodies a subset of network functions attributed to the respective VNF. The cost of the deployment is calculated in terms of the DC infrastructure resources, such as computation and networking. On the other hand, the radio resource abstraction and slicing need to be carefully consideration, especially for the virtualization in CC-RANs, which involves the radio resource sharing and the protocol design. What’s more, virtualization for cloud computing brings in special challenges for cloud service performance [112], and the state of the art of cloud service performance evaluation has been presented from the system modeling perspective in [112]. The performance metrics will be more diversified, and the traditional system models and algorithms may be ineffective. Therefore, performance analysis urgently needs to be well dealt with not only for confirming the performance metrics, but also for the system models and algorithms according to different design and optimization problems.

6.4 Network Security for Virtualized CC-RANs

Due to the virtualization technology applying to CC-RANs, the extended radio spectrum sharing and infrastructure virtualization will let CC-RANs more flexible and open, which may be easily attacked. At the same time, a large number of intelligent IoT devices/nodes, even including malicious pseudo basestations, may access networks with well self-adaption/context awareness capacities, which will bring lots of potential threats to network security. These threats may not only harm the service and information security of some users, but also restrict the entire network capacity. What´s more, the encryption often needs considerable computational resources and communication overheads [42]. In these contexts, traditional authentication, authorization, and even accounting strategies may need to be redesigned to satisfy diverse radio accesses and dynamic virtualized CC-RAN reconstruction.

7 Conclusions

This paper simply surveyed the state-of-the-art virtualization technology in CC-RANs, mainly concentrating on C-RAN, H-CRAN and F-RAN. The background of network virtualization, virtualization architecture in different CC-RANs, virtualization key enabling technologies, and open issues were presented. Since wireless network virtualization has attracted much attention, the benefits of virtualization, the diverse radio access demands and cost-efficient requirements render the introduction of virtualization technology into CC-RANs as an overwhelming trend. Meanwhile, the key enabling technologies for CC-RAN virtualization are summarized as virtual resource allocation, RAN slicing, mobility management, and social-awareness for satisfying the requirements of isolation, customization and high radio resource utilization. However, given the relative infancy of the CC-RAN virtualization, there are quite a number of outstanding problems that needs further investigation from the perspective of promising key technologies and advanced solutions. At last, considering the deployment of virtualized CC-RANs, we mainly discussed the virtualization levels, signaling design, performance analysis, and network security as the challenges and open issues. Besides, the advanced isolation technologies among multiple virtual CC-RANs networks, the network intelligence of CC-RAN virtualization, as well as the practical deployment schemes are also urgent to be discussed and further investigated.

References

[1] M. Peng, Y. Li, Z. Zhao, and C. Wang, "System architecture and key technologies for 5G heterogeneous cloud radio access networks," *IEEE Network*, vol. 29, no. 2, pp. 6–14, Mar. 2015. doi: 10.1109/MNET.2015.7064897.

[2] M. Peng and W. Wang, "Technologies and standards for TD-SCDMA evolutions to IMT-Advanced," *IEEE Communications Magazine*, vol. 47, no. 12, pp. 50–58, Dec. 2009. doi: 10.1109/MCOM.2009.5350368.

[3] B. Cao, F. He, Y. Li, C. Wang, and W. Liang, "Software defined virtual wireless network: Framework and challenges," *IEEE Network*, vol. 29, no.4, pp.6–12, Jul.-Aug. 2015. doi: 10.1109/MNET.2015.7166185.

[4] H. Zhang, S. Vreic, G. Senarath, et al., "5G wireless network: MyNET and SONAC," *IEEE Network*, vol. 29, no. 4, pp. 14–23, Jul.-Aug. 2015. doi:10.1109/MNET.2015.7166186.

[5] S. Dixit, C. Politis, and A. T. Papathanassiou, "Network virtualization in the wireless world," *IEEE Vehicular Technology Magazine*, vol.10, no. 3, pp. 27–29, Sept. 2015. doi:10.1109/MVT.2015.2446452.

[6] K. Zhu and E. Hossain, "Virtualization of 5G cellular networks as a hierarchical combinatorial auction," *IEEE Transaction on Mobile Computing*, vol. 15, no. 10, pp. 2640–2654, Oct. 2016. doi:10.1109/TMC.2015.2506578.

[7] M. M. Rahman, C. Despins, and S. Affes, "Design optimization of wireless access virtualization based on cost and QoS trade-off utility maximization," *IEEE Transactions on Wireless Communications*, vol. 15, no. 9, pp. 6146–6162, Sept. 2016. doi: 10.1109/TWC.2016.2580505.

[8] R. Riggio, A. Bradai, D. Harutyunyan, T. Rasheed, and T. Ahmed, "Scheduling wireless virtual networks functions," *IEEE Transactions on Network and Service Management*, vol. 13, no. 2, pp. 240–252, Jun. 2016. doi: 10.1109/TNSM.2016.2549563.

[9] Y. Li and M. Chen, "Software-defined network function virtualization: a survey," *IEEE Access*, vol. 3, pp. 2542–2553, Dec. 2015. doi: 10.1109/ACCESS.2015.2499271.

[10] C. Liang and F. R. Yu, "Wireless virtualization for next generation mobile cellular networks," *IEEE Wireless Communications*, vol. 22, no. 1, pp. 61–69, Feb. 2015. doi: 10.1109/MWC.2015.7054720.

[11] H. Huang, P. Li, S. Guo, and W. Zhuang "Software-defined wireless mesh networks: architecture and traffic orchestration," *IEEE network*, vol. 29, no. 4, pp. 24–30, Jul./Aug. 2015. doi: 10.1109/MNET.2015.7166187.

[12] T. Wood, K. K. Ramakrishnan, J. Hwang, G. Liu, and W. Zhang, "Toward a software-based network integrating software defined networking and network function virtualization," *IEEE Network*, vol. 29, no. 3, pp. 36–41, May-Jun. 2015. doi: 10.1109/MNET.2015.7113223.

[13] J. G. Herrera and J. F. Botero, "Resource allocation in NFV: a comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, Sept. 2016. doi: 10.1109/TNSM.2016.2598420.

[14] Q. Zhou, C. Wang, S. Mclaughlin, and X. Zhou, "Network virtualization and resource description in software-defined wireless networks," *IEEE Communications Magazine*, vol. 53, no. 11, pp. 110–117, Nov. 2015. doi: 10.1109/MCOM.2015.7321979.

[15] Z. Feng, C. Qiu, Z. Feng, et al., "An effective approach to 5G: wireless network virtualization," *IEEE Communications Magazine*, vol.53, no.12, pp. 53–59, Dec. 2015. doi: 10.1109/MCOM.2015.7355585.

[16] C. J. Bernardos, A. D. L. Olive, P. Serrano, et al., "An architecture for software defined wireless networking," *IEEE Wireless Communications*, vol. 21, no. 3, pp. 52–61, Jun. 2014. doi: 10.1109/MWC.2014.6845049.

[17] H. Li, M. Dong, and K. Ota, "Radio access network virtualization for the social internet of things," *IEEE Could Computing*, vol. 2, no. 6, pp. 42–50, Nov.-Dec. 2015. doi: 10.1109/MCC.2015.114.

[18] F. Granelli, A. A. Gebremariam, M. Usman, et al., "Software defined and virtualized wireless access in future wireless networks: scenarios and standards," *IEEE Communications Magazine*, vol. 53, no. 6, pp. 26–34, Jun. 2015. doi: 10.1109/MCOM.2015.7120042.

[19] X. Costa-Prez, J. Swetina, T. Guo, R. Mahindra, and S. Rangarajan, "Radio access network for future mobile carrier networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 27–35, Jul. 2013. doi: 10.1109/MCOM.2013.6553675.

[20] R. Kokku, R. Mahindra, H. Zhang, and S. Ranfarajan, "NVS: a substrate for virtualizing wireless resources in cellular networks," *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, pp. 1333–1346, Oct. 2012. doi: 10.1109/TNET.2011.2179063.

[21] G. Tseliou, F. Adelantado, and C. Verikoukis, "Scalable RAN virtualization in multitenant LTE-A heterogeneous networks," *IEEE Transactions on Vehicular Technology*, vol. 65, pp. 6651–6664, Aug. 2016. doi:10.1109/TVT.2015.2475641.

[22] C. Liang and F. R. Yu, "Wireless network virtualization: a survey, some research issues and challenges," *IEEE Communications Surveys & Tutorials*, vol.17, no. 1, pp. 358–380, first quarter 2015. doi: 10.1109/COMST.2014.2352118.

[23] F. Callegati, W. Cerroni, C. Contoli, et al., "SDN for dynamic NFV deployment," *IEEE Communications Magazine*, vol. 54, no. 10, pp. 89–95, Oct. 2016. doi: 10.1109/MCOM.2016.7588275.

[24] VMWARE. (2014, Jul.). Virtualization essentials [Online]. Available: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/ebook/gated-vmw-ebook-virtualizationessentials.pdf

[25] H. Hawilo, A. shami, M. Mirahmadi, and R. Asal, "NFV: state of the art, challenges, and implementation in next generation mobile networks (vEPC)," *IEEE Network.*, vol. 28, no. 6, pp. 18–26, Nov.-Dec. 2014. doi: 10.1109/MNET.2014.6963800.

[26] ETSI. (2012, Oct.). Network function virtualization: an introduction, benefits, enablers, challenges, & call for action, *SDN and OpenFlow World Congress* [Online]: Available: https://portal.etsi.org/nfv/nfv-whitepaper. pdf

[27] A. Wang, M. Iyer, R. Dutta, G. N. Rouskas, and I. Baldine, "Network virtualization: technologies, perspectives, and frontiers," *Journal of Lightwave Technology*, vol. 31, no. 4, pp. 523–537, Feb. 2013. doi: 10.1109/JLT.2012.2213796.

[28] M. A. Marotta, N. Kaminski, I. Gomez-Miguelez, et al., "Resource sharing in heterogeneous cloud radio access networks," *IEEE Wireless Communications*, vol. 22, no. 3, pp. 74–82, Jun. 2015. doi: 10.1109/MWC.2015.7143329.

[29] C. Liang, F. R. Yu, and X. Zhang, "Information-centric network function virtualization over 5G mobile wireless networks," *IEEE Network.*, vol. 29, no. 3, pp. 68–74, May-Jun. 2015. doi: 10.1109/MNET.2015.7113228.

[30] M. Yang, Y. Li, D. Jin, et al., "Software-defined and virtualized future and wireless networks: a survey," *Mobile Networking and Application*, vol. 20, no. 1, pp. 4–18, Feb. 2015. doi:10.1007/s11036-014-0533-8.

[31] P. Rost, I. Berberana, A. Maeder, et al., "Benefits and challenges of virtualization in 5G radio access networks," *IEEE Communications Magazine*, vol.53, no. 12, pp. 75–82, Dec. 2015. doi: 10.1109/MCOM.2015.7355588.

[32] N. Marshall and J. Hoffman. (2015, Jul.). NFV: radio virtualization in the RAN [Online]. Available: https://www.abiresearch.com

[33] M. Y. Arslan, K. Sundaresan, and S. Rangarajan, "Software-defined networking in cellular radio access networks: potential and challenges," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 150–156, Jan. 2015. doi: 10.1109/MCOM.2015.7010528.

[34] EU FP7. (2017, Mar.). FLAVIA project [Online]. Available: http://www.ict-flavia.eu

[35] Small Cell Forum. (2015, Jun.). Virtualization for small cells: overview, Rep. version:106.07.01 [Online]. Available: http://scf.io/en/documents/106-Virtualization0-for-small-cells-Overview.php

[36] 5GPPP 5GEx project, "5GEx Multi-domain service creation-from 90 days to 90

minutes," A 5GEx White Paper V1, Mar. 2016.
[37] 5GPPP. (2017, Mar.). SEAME project [Online]. Available: https://5g-ppp.eu/sesame
[38] X. Zhou, R. Li, T. Chen, and H. Zhang, "Network slicing as a service: enabling enterprises' own software-defined cellular networks," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 146–153, Jul. 2016. doi: 10.1109/MCOM.2016.7509393.
[39] D. Sabella, P. Rost, Y. Sheng et al., "RAN as a service: challenges of designing a flexible RAN architecture in a cloud-based heterogeneous mobile network," in *Proc. Future Network & Mobile Summit*, Lisboa, Portugal, 2013, pp. 1–8.
[40] M. Patwary, S. K. Sharma, S. Chatzinotas, et al., "Universal intelligent small cell (UnISCell) for next generation cellular networks," Digital Communications and Networks, vol. 2, no. 4, Nov. 2016, pp. 167–174, doi: 10.1016/j.dcan.2016.09.003.
[41] R. P. Esteves, L. Z. Granville, and R. Boutaba, "On the management of virtual networks," *IEEE Communications Magazine.*, vol. 51, no. 7, pp. 80–88, Jul. 2014. doi: 10.1109/MCOM.2013.6553682.
[42] M. Peng, Y. Sun, X. Li, Z. Mao, and C. Wang, "Recent advances in cloud radio access networks: system architecture, key techniques, and open issues," *IEEE Communications Surveys &Tutorials*, vol. 18, no. 3, pp. 2282–2308, third quarter 2016. doi: 10.1109/COMST.2016.2548658.
[43] A. Checko, H. L. Christiansen, Y. Yan, et al., "Cloud RAN for mobile networks-a technology overview," *IEEE Communications Surveys &Tutorials*, vol. 17, no. 1, pp. 405–426, first quarter 2015. doi: 10.1109/COMST.2014.2355255.
[44] S. Abdelwahab, B. Hamdaoui, M. Guizani, and T. Znati, "Network function virtualization in 5G," *IEEE Communications Magazine*, vol. 54, no. 4, pp. 84–91, Apr. 2016. doi: 10.1109/MCOM.2016.7452271.
[45] M. I. M. Alfaqawi, J. Chebil, M. H. Habaebi, and D. Datla, "Wireless distributed computing for cyclostationary feature detection," *Digital Communications and Networks*, vol. 2, no. 1, pp. 47–56, Feb. 2016. doi: 10.1016/j.dcan.2015.09.003.
[46] X. Li, X. Wang, C. Zhu, W. Cai, and V. C. M. Leung, "Caching-as-a-service: Virtual caching framework in the cloud-based mobile networks," in *IEEE INFOCOM 2015 Workshops on Computer Communications (INFOCOM WKSHPS)*, Hong Kong, China, pp. 372–377, Aug. 2015. doi: 10.1109/INFCOMW.2015.7179413.
[47] S. Buzzi, C.-L. I, T. E. Klein, et al., "A survey of energy-efficient techniques for 5G networks and challenges Ahead," *IEEE Journal on Selected Areas in Communications.*, vol. 34, no. 4, pp. 697–709, Apr. 2016. doi: 10.1109/JSAC.2016.2550338.
[48] ITU, "IMT vision—framework and overall objectives of the future development of IMT for 2020 and beyond," ITU-R M.2083-0, Sept. 2015.
[49] Z. Chang, Z. Zhou, S. Zhou, T. Ristaniemi, and Z. Niu. (2016, Apr.). Towards service-oriented 5G: virtualizing the networks for everything-as-a-service [Online]. Available: https://arxiv.org/pdf/1604.01739
[50] S. Barbarossa, S. Sardellitti, and P. D. Lorenzo, "Computing while computing: distributed mobile cloud computing over 5G heterogeneous networks," *IEEE Signal Processing Magazine*, vol. 31, no. 6, pp. 45–55, Nov. 2014. doi: 10.1109/MSP.2014.2334709.
[51] M. Peng, Y. Yu, H. Xiang, and H. V. Poor, "Energy-efficient resource allocation optimization for multimedia heterogeneous cloud radio access networks", *IEEE Trans. Multimedia*, vol. 18, no. 5, pp. 879–892, May 2016.
[52] S. Zhou, J. Gong, Z. Zhou, W. Chen, and Z. Niu, "GreenDelivery: proactive content caching and push with energy-harvesting based small cells" *IEEE Communications Magazine*, vol. 53, no. 4, pp. 142–149, Apr. 2015. doi: 10.1109/MCOM.2015.7081087.
[53] R. Khanna, H. Liu, and T. Rangarajan, "Wireless data center management: sensor network applications and challenges," *IEEE Microwave Magazine*, vol. 15, no. 7, pp. S45–S60, Nov./Dec. 2014. doi: 10.1109/MMM.2014.2356151.
[54] E. S. Correa, L. A. Fletscher, and J. F. Botero, "Virtual data center embedding: a survey," *IEEE Latin America Transactions*, vol. 13, no. 5, pp. 1661–1670, May 2015. doi: 10.1109/TLA.2015.7112029.
[55] Techopedia. (2017, Mar.). Data center virtualization [Online]. Available: https://www.techopedia.com/definition/29883/data-center-virtualization
[56] H. Shirayanagi, H. Yamada, and K. Kono, "Honeyguide: a VM migration-aware network topology for saving energy consumption in data center networks," in *IEEE Symposium Computers and Communications (ISCC)*, Cappadocia, Turkey, 2012. pp. 000460–000467. doi: 10.1109/ISCC.2012.6249339.
[57] S. Zhou, T. Zhao, Z. Niu, and S. Zhou, "Software-defined hyper-cellular architecture for green and elastic wireless access," *IEEE Communications Magazine*, vol. 54, no. 1, Jan. 2016. doi: 10.1109/MCOM.2016.7378420.
[58] Z. Zaidi, V. Friderikos, and M. A. Imran, "Future RAN architecture: SD-RAN through a general-purpose processing platform," *IEEE Vehicular Technology Magazine*, vol. 10, no. 1, pp. 52–60, Mar. 2015. doi: 10.1109/MVT.2014.2380632.
[59] C.-L. I, Y. Yuan, J. Huang, et al., "Rethink fronthaul for soft RAN," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 82–88, Sept. 2015. doi: 10.1109/MCOM.2015.7263350.
[60] T. Chen, M. Matinmikko, X. Chen, X. Zhou, and P. Ahokangas, "Software defined mobile networks: concept, survey, and research directions," *IEEE Communications Magazine*, vol. 53, no. 11, pp. 126–133, Nov. 2015. doi: 10.1109/MCOM.2015.7321981.
[61] M. Y. Arslan, K. Sundaresan, and S. Rangarajan, "Software-defined networking in cellular radio access networks: potential and challenges," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 150–156, Jan. 2015. doi: 10.1109/MCOM.2015.7010528.
[62] A. M. Akhtar, X. wang, and L. Hanzo, "Synergistic spectrum sharing in 5G HetNets: a harmonized SDN-enabled approach," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 40–47, Jan. 2016. doi: 10.1109/MCOM.2016.7378424.
[63] A. Gupta and R. K. Jha, "A survey of 5G network: architecture and emerging technologies," *IEEE Access*, vol. 3, pp. 1206–1232, Jul. 2015. doi: 10.1109/ACCESS.2015.2461602.
[64] R. Riggio, M. K. Marina, J. Schulz-Zander, S. Kuklinski, and T. Rashed, "Programming abstractions for software-defined wireless networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 2, pp. 146–162, Jun. 2015. doi: 10.1109/TNSM.2015.2417772.
[65] M. Peng, C. Wang, J. Li, H. Xiang, and V. Lau, "Recent advances in underlay heterogeneous networks: interference control, resource allocation, and self-organization," *IEEE Communications on Surveys & Tutorials.*, vol. 17, no. 2, pp. 700–729, Second quarter 2015. doi: 10.1109/COMST.2015.2416772.
[66] H. Ibrahim, H. ElSawy, U. T. Nguyen, and M.-S. Alouini, "Mobility-aware modeling and analysis of dense cellular networks with C-Plane/U-Plane split architecture," *IEEE Transactions on Communications.*, vol. 64, no. 11, pp. 4879–4894, Nov. 2016. doi: 10.1109/TCOMM.2016.2609905.
[67] M. Peng, Y. Li, Z. Zhao, J. Jiang, J. Li, and C. Wang, "Heterogeneous cloud radio access networks: a new perspective for enhancing spectral and energy efficiencies," *IEEE Wireless Communications*, vol. 21, no. 6, pp. 126–135, Dec. 2014. doi: 10.1109/MWC.2014.7000980/ISSN: 1536-1284.
[68] M. Armbrust, A. Fox, R. Griffith, et al., "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010. doi: 10.1145/1721654.1721672.
[69] M. Peng, C. Wang, V. Lau, and H. V. Poor, "Fronthaul-constrained cloud radio access networks: insights and challenges," *IEEE Wireless Communications.*, vol. 22, no. 2, pp. 152–160, Apr. 2015. doi: 10.1109/MWC.2015.7096298.
[70] O. Simeone, A. Maeder, M. Peng, O. Sahin, and W. Yu, "Cloud radio access network: virtualizing wireless access for dense heterogeneous systems," *Journal of Communications and Networks*, vol. 18, no. 2, pp. 135–149, Apr. 2016. doi: 10.1109/JCN.2016.000023.
[71] K. A. Meerja, A. Shami, and S. Refaey, "Hailing cloud empowered radio access networks," *IEEE Wireless Communications*, vol. 22, no. 1, pp. 122–129, Feb. 2015. doi: 10.1109/MWC.2015.7054727.
[72] A. Blenk, A. Basta, M. Reisslein, and W.Kellerer, "Survey on network virtualization hypervisors for software defined networking," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 655–685, First quarter 2016. doi: 10.1109/COMST.2015.2489183.
[73] C.-L. I, J. Huang, R. Duan, et al., "Recent progress on C-RAN centralization and cloudification," *IEEE Access*, vol. 2, pp. 1030–1039, Aug. 2014. doi: 10.1109/ACCESS.2014.2351411.
[74] C.-L. I, C. Rowell, S. Han, et al., "Toward green and soft: a 5G perspective," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 66–73, Feb. 2014. doi: 10.1109/MCOM.2014.6736745.
[75] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and it's role in the internet of things," in *Proc. First Edition of MCC Workshop on the Mobile Cloud Computing*, Helsinki, Finland, 2012, pp. 13–16.
[76] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: a platform for internet of things and analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments, N. Bessis and C. Dobre ed*. New York, USA: Springer, 2014, vol. 546, pp. 169–186. doi: 10.1007/978-3-319-05029-4_7.
[77] Y. Ai, M. Peng, and K. Zhang. (2017). Edge cloud computing technologies for internet of things: a primer, *Digital Communications and Networks* [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352864817301335
[78] K. Liang, L. Zhao, X. Chu, and H.-H. Chen, "An integrated architecture for software defined and virtualized radio access networks with fog computing,"

IEEE Network, vol. 31, no. 1, pp. 80–87, Jan./Feb. 2017. doi: 10.1109/MNET.2017.1600027NM.

[79] M. Peng, S. Yan, K. Zhang, and C. Wang, "Fog-computing-based radio access networks: issues and challenges," *IEEE Network*, vol. 30, no. 4, pp. 46–53, Jul.-Aug. 2016. doi: 10.1109/MNET.2016.7513863.

[80] M. Peng, and K. Zhang, "Recent advances in fog radio access networks: performance analysis and radio resource allocation," *IEEE Access*, vol. 4, pp. 5003–5009, Aug. 2016. doi: 10.1109/ACCESS.2016.2603996.

[81] S.-C. Hung, H. Hsu, S. -Y. Lien, and K. -C. Chen, "Architecture harmonization between cloud radio access networks and fog network," *IEEE Access*, vol. 3, pp. 3019–3034, Dec. 2015. doi: 10.1109/ACCESS.2015.2509638.

[82] R. Tandon and O. Simeone "Harnessing cloud and edge synergies: toward an information theory of fog radio access networks," *IEEE Communications Magazine*, vol. 54, no. 8, pp. 44–50, Aug. 2016. doi: 10.1109/MCOM.2016.7537176.

[83] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the suitability of fog computing in the context of internet of things," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–14, Oct. 2015. doi:10.1109/TCC.2015.2485206.

[84] S. H. Yeganeh and Y. Ganjali, "Kandoo: a framework for efficient and scalable offloading of control applications," in *Proc. 1st Workshop on Hot Topics in Software Defined Networks*, New York, USA, 2012, pp. 19–24.

[85] Nokia, "Dynamic end-to-end network slicing for 5G," Whitepaper, Jun. 2016.

[86] 5GPPP, "Deliverable D2.2: draft overall 5G RAN design," Whitepaper, Jun. 2016.

[87] ZTE Corporation, "Some Clarification for NW Slicing," (Type: discussion), Reno, US, R3-162723, Nov. 2016.

[88] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: a survey," IEEE *Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 1888–1906, FourthQuarter 2013. doi: 10.1109/SURV.2013.013013.00155.

[89] T. LeAnh, N. H. Trane, D. T. Ngo, and C. S. Hong, "Resource allocation for virtualized wireless networks with backhaul constraints," *IEEE Communications Letters*, Oct. 2016, doi: 10.1109/LCOMM.2016.2617307.

[90] V. Jumba, S. Parsaeefard, M. Derakhshani, and T. Le-Ngoc, "Resource provisioning in wireless virtualized networks via massive-MIMO," *IEEE Wireless Communications Letters*, vol. 4, no. 3, pp.237–240, Jun. 2015. doi: 10.1109/LWC.2015.2402126.

[91] S. Parsaeefard, R. Dawadi, M. Derakhshani, and T. Le-Ngoc, "Joint user-association and resource-allocation in virtualized wireless networks," *IEEE Access*, Apr. 2016, doi: 10.1109/ACCESS.2016.2560218.

[92] F. Fu and U. C. Kozat, "Stochastic game for wireless network virtualization," *IEEE/ACM Transactions on Networking*, vol. 21, no. 1, pp. 84–97, Feb. 2013. doi: 10.1109/TNET.2012.2190419.

[93] E. Amaldi, S. Coniglio, A. M. C. A. Koster, and M. Tieves, "On the computational complexity of the virtual network embedding problem," *Electronic Notes in Discrete Mathematics*, vol. 52, pp. 213–220, Jun. 2016. doi: 10.1016/j.endm.2016.03.028.

[94] F. Z. Yousaf and T. Taleb, "Fine-grained resource-aware virtual network function management for 5G carrier cloud," *IEEE Network*, vol. 30, no. 2, pp. 110–115, Mar.-Apr. 2016. doi: 10.1109/MNET.2016.7437032.

[95] S.-H. Park, O. Simeone, and S. Shamai(Shitz), "Joint optimization of cloud and edge processing for fog radio access networks" *IEEE Transactions on Wireless Communications*, vol. 15, no. 11, pp. 7621–7632, Nov. 2016. doi: 10.1109/TWC.2016.2605104.

[96] NGMN Alliance, "Description of network slicing concept," Final Deliverable (approved), January 2016.

[97] O. Sallent, J. Perez-Romero, R. Ferrus, and R. Agusti, "On radio access network slicing from a radio resource management perspective," *IEEE Wireless Communications*, vol. PP, no. 99, pp. 2–10, Apr. 2017. doi: 10.1109/MWC.2017.1600220WC.

[98] A. Ksentini and N. Nikaein, "Toward enforcing network slicing on RAN: flexibility and resource abstraction," *IEEE Communications Magazine.*, vol. 55, no. 6, pp. 102–108, June 2017. doi: 10.1109/MCOM.2017.1601119.

[99] X. Foukas, N. Nikaein, and M. M. Kassem, "FlexRAN: a flexible and programmable platform for software-defined radio access networks," in *Proc. 12th ACM International Conference on Emerging Networking Experiments and Technologies*, Irvine, USA, Dec. 2016, pp. 427–441.

[100] China Mobile Communications Corporation, Huawei Technologies Co. Ltd., Deutsche Telekom AG, and Volkswagen, "5G service-guaranteed networking slicing white paper," White Paper V1.0, Feb. 2017.

[101] M. Richart, J. Baliosian, J. Serrat, and J. Gorricho, "Resource slicing in virtual wireless network: a survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 462–476, Sept. 2016. doi: 10.1109/TNSM.2016.2597295.

[102] F. R. Yu, V. W. S. Wong, J.-H. Song, V. C. M. Leung, and H. C. B. Chan. (2011, Apr.). Next generation mobility management: an introduction, *Wireless Communications on Mobile Computing* [Online]. *11(4)*, pp. 446–458. Available: http://onlinelibrary.wiley.com/doi/10.1002/wcm.904/pdf

[103] L. Zhu, F. R. Yu, T. Tang, and B. Ning, "Handoff performance improvement in an integrated train-ground communication system based on wireless network virtualization," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1165–1178, May 2017. doi: 10.1109/TITS.2016.2620171.

[104] X. Duan and X. Wang, "Authentication handover and privacy protection in 5G HetNets using software-defined networking," *IEEE Communications Magazine.*, vol. 53, no. 4, pp. 28–35, Apr. 2015. doi: 10.1109/MCOM.2015.7081072.

[105] H. Zhang, N. Liu, X. Chu, et al. (2017, Apr.). Network slicing based 5G and future mobile networks mobility, resource management, and challenges [Online]. Available: https:// arxiv.org/abs/1704.07038v1

[106] F. Xia, L. Liu, J. Ma, and A. V. Vasilakos, "Socially aware networking: a survey," *IEEE System Journal*, vol. 9, no. 3, pp. 904–921, Sept. 2015. doi: 10.1109/JSYST.2013.2281262.

[107] Y. Cao, C. Long, T. Jiang, and S. Mao, "Share communication and computation resources on mobile devices: a social awareness perspective," *IEEE Wireless Communications*, vol. 23, no. 4, pp. 52–59, Aug. 2016. doi: 10.1109/MWC.2016.7553026.

[108] L. Jiang, H. Tian, Z. Xing, et al., "Social-aware energy harvesting device-to-device communications in 5G networks," *IEEE Wireless Communications*, vol. 23, no. 4, pp. 20–27, Aug. 2016. doi: 10.1109/MWC.2016.7553022.

[109] B. Bai, L. Wang, Z. Han, W. Chen, and T. Svensson, "Caching based socially-D2D communications in wireless content delivery networks: a hypergraph framework," *IEEE Wireless Communications.*, vol. 23, no.4, pp. 74–81, Aug. 2016. doi: 10.1109/MWC.2016.7553029.

[110] A. Leivadeas, C. Papagianni, and S. Papavassiliou, "Socio-aware virtual network embedding," *IEEE Network*, vol. 26, no. 5, pp. 35–43, Sept.-Oct. 2012. doi: 10.1109/MNET.2012.6308073.

[111] F. Z. Yousaf, P. Loureiro, F. Zdarsky, T. Taleb, and M. Liebsch, "Cost analysis of initial deployment strategies for virtualized mobile core network functions," *IEEE Communications Magazine*, vol. 53, no. 12, pp. 60–66, Dec. 2015. doi: 10.1109/MCOM.2015.7355586.

[112] Q. Duan, "Cloud service performance evaluation: status, challenges, and opportunities-a survey from the system modeling perspective," *Digital Communications and Networks*, vol. 3, no. 2, pp. 101–111, May 2017. doi.org/10.1016/j.dcan.2016.12.002.

Manuscript received: 2017-07-14

Biographies

ZHANG Xian (zhangxian_email@163.com) received the B.E. degree in telecommunications engineering from Chongqing University of Posts and Telecommunications, China in 2016. He is currently pursuing Ph.D. with Key Laboratory of Universal Wireless Communication, Ministry of Education, Beijing University of Posts and Telecommunications (BUPT), China. His research interests include resources allocation and optimization for fog computing based radio access networks (F-RANs) and software-defined radio access networks.

PENG Mugen (pmg@bupt.edu.cn) received the Ph.D. degree in communication and information systems from Beijing University of Posts and Telecommunications (BUPT), China, in 2005. Afterward, he joined BUPT, where he has been a full professor with the School of Information and Communication Engineering since 2012. His main research interests focus on cooperative communications, self-organization networking, heterogeneous networking, cloud communication, and the Internet of Things. He has authored/coauthored over 70 refereed IEEE journal papers and over 200 conference proceeding papers. He was a recipient of the 2014 IEEE ComSoc AP Outstanding Young Researcher Award, and the best paper award in JCN, IEEE WCNC 2015, WASA 2015, GameNets 2014, IEEE CIT 2014, ICCTA 2011, IC-BNMT 2010, and IET CCWMC 2009. He is on the Editorial/Associate Editorial Board of *IEEE Communications Magazine*, *IEEE Access*, *IEEE Internet of Things Journal*, *IET Communications*, and *China Communications*. He is the Fellow of IET.

ZTE Communications Guidelines for Authors

- **Remit of Journal**

ZTE Communications publishes original theoretical papers, research findings, and surveys on a broad range of communications topics, including communications and information system design, optical fiber and electro-optical engineering, microwave technology, radio wave propagation, antenna engineering, electromagnetics, signal and image processing, and power engineering. The journal is designed to be an integrated forum for university academics and industry researchers from around the world.

- **Manuscript Preparation**

Manuscripts must be typed in English and submitted electronically in MS Word (or compatible) format. The word length is approximately 3000 to 8000, and no more than 8 figures or tables should be included. Authors are requested to submit mathematical material and graphics in an editable format.

- **Abstract and Keywords**

Each manuscript must include an abstract of approximately 150 words written as a single paragraph. The abstract should not include mathematics or references and should not be repeated verbatim in the introduction. The abstract should be a self-contained overview of the aims, methods, experimental results, and significance of research outlined in the paper. Five carefully chosen keywords must be provided with the abstract.

- **References**

Manuscripts must be referenced at a level that conforms to international academic standards. All references must be numbered sequentially in-text and listed in corresponding order at the end of the paper. References that are not cited in-text should not be included in the reference list. References must be complete and formatted according to ZTE Communications Editorial Style. A minimum of 10 references should be provided. Footnotes should be avoided or kept to a minimum.

- **Copyright and Declaration**

Authors are responsible for obtaining permission to reproduce any material for which they do not hold copyright. Permission to reproduce any part of this publication for commercial use must be obtained in advance from the editorial office of *ZTE Communications*. Authors agree that a) the manuscript is a product of research conducted by themselves and the stated co-authors, b) the manuscript has not been published elsewhere in its submitted form, c) the manuscript is not currently being considered for publication elsewhere. If the paper is an adaptation of a speech or presentation, acknowledgement of this is required within the paper. The number of co-authors should not exceed five.

- **Content and Structure**

ZTE Communications seeks to publish original content that may build on existing literature in any field of communications. Authors should not dedicate a disproportionate amount of a paper to fundamental background, historical overviews, or chronologies that may be sufficiently dealt with by references. Authors are also requested to avoid the overuse of bullet points when structuring papers. The conclusion should include a commentary on the significance/future implications of the research as well as an overview of the material presented.

- **Peer Review and Editing**

All manuscripts will be subject to a two-stage anonymous peer review as well as copyediting, and formatting. Authors may be asked to revise parts of a manuscript prior to publication.

- **Biographical Information**

All authors are requested to provide a brief biography (approx. 100 words) that includes email address, educational background, career experience, research interests, awards, and publications.

- **Acknowledgements and Funding**

A manuscript based on funded research must clearly state the program name, funding body, and grant number. Individuals who contributed to the manuscript should be acknowledged in a brief statement.

- **Address for Submission**

http://mc03.manuscriptcentral.com/ztecom
12F Kaixuan Building, 329 Jinzhai Rd, Hefei 230061, P. R. China

ZTE COMMUNICATIONS

ZTE Communications has been indexed in the following databases:

- Abstract Journal
- Cambridge Scientific Abstracts (CSA)
- China Science and Technology Journal Database
- Chinese Journal Fulltext Databases
- Inspec
- Ulrich's Periodicals Directory
- Wanfang Data—Digital Periodicals

ZTE COMMUNICATIONS
Vol. 15 No. 4 (Issue 59)
Quarterly
First English Issue Published in 2003

Supervised by:
Anhui Science and Technology Department

Sponsored by:
Anhui Science and Technology Information Research Institute and ZTE Corporation

Staff Members:
Editor-in-Chief: CHEN Jie
Executive Associate Editor-in-Chief: HUANG Xinming
Editor-in-Charge: ZHU Li
Editors: XU Ye, LU Dan, ZHAO Lu
Producer: YU Gang
Circulation Executive: WANG Pingping
Assistant: WANG Kun

Editorial Correspondence:
Add: 12F Kaixuan Building, 329 Jinzhai Road,
Hefei 230061, P. R. China
Tel: +86-551-65533356
Fax: +86-551-65850139
Email: magazine@zte.com.cn

Published and Circulated (Home and Abroad) by:
Editorial Office of *ZTE Communications*

Printed by:
Hefei Tiancai Color Printing Company

Publication Date: October 25, 2017

Publication Licenses:
ISSN 1673-5188
CN 34-1294/ TN

Annual Subscription: RMB 80