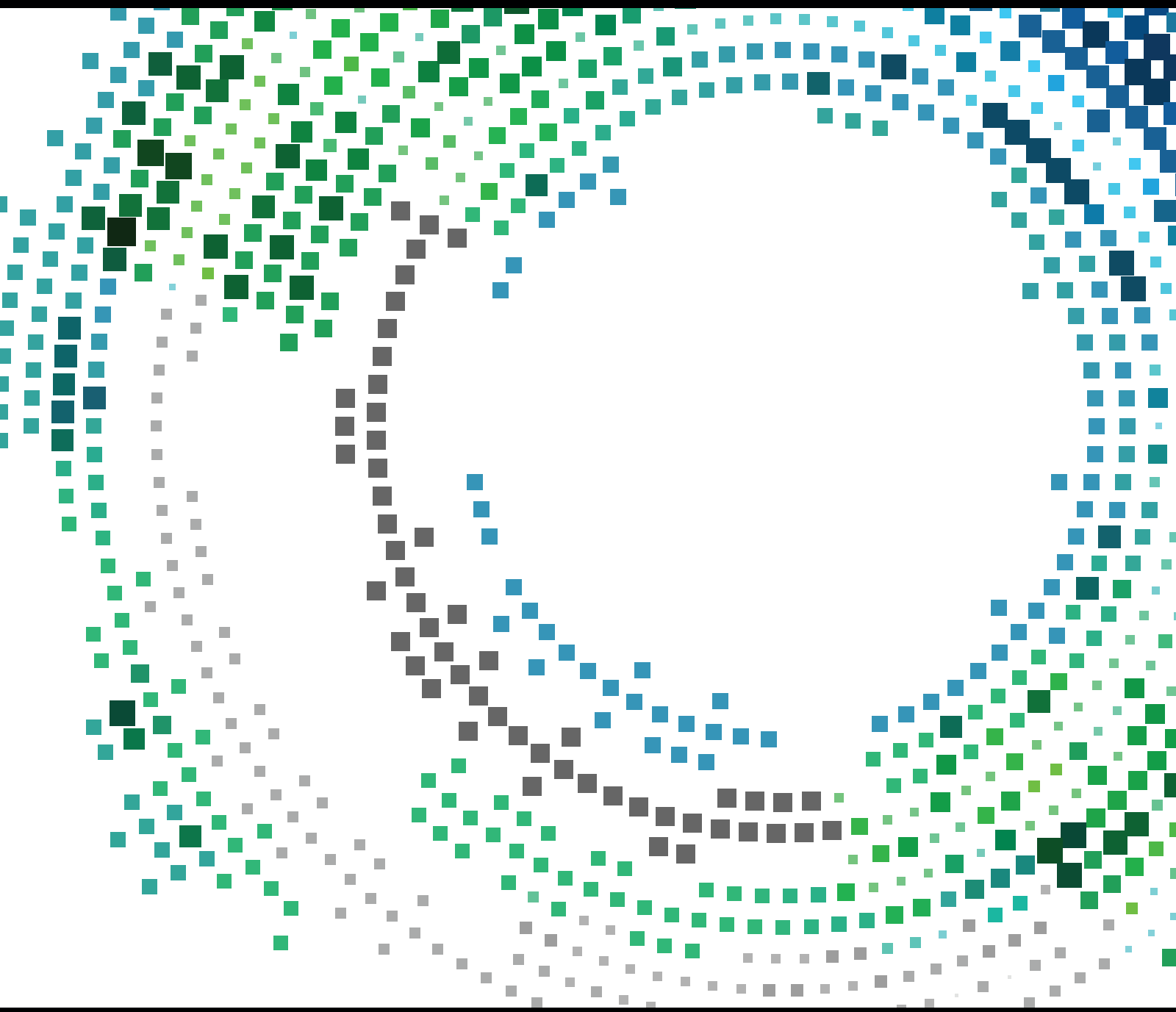
Mobile Information Systems

Sustainable Mobile Computing and Communications

Lead Guest Editor: Karl Andersson
Guest Editors: Eric Rondeau, Ah-Lian Kor, and Dan Johansson



Sustainable Mobile Computing and Communications

Mobile Information Systems

Sustainable Mobile Computing and Communications

Lead Guest Editor: Karl Andersson
Guest Editors: Eric Rondeau, Ah-Lian Kor, and Dan Johansson

Editorial Board

Contents

Hindawi
Mobile Information Systems
Volume 2017, Article ID 1098264, 2 pages
https://doi.org/10.1155/2017/1098264

Editorial
Sustainable Mobile Computing and Communications

Karl Andersson,[1] Eric Rondeau,[2] Ah-Lian Kor,[3] and Dan Johansson[4]

[1] *Pervasive and Mobile Computing Laboratory, Luleå University of Technology, 931 87 Skellefteå, Sweden*
[2] *Université de Lorraine, BP 70239, 54506 Vandoeuvre-lès-Nancy, France*
[3] *School of Computing, Creative Technologies and Engineering, Leeds Beckett University, Headingley Campus, Leeds LS6 3QS, UK*
[4] *Department of Informatics, Umeå University, 901 87 Umeå, Sweden*

Correspondence should be addressed to Karl Andersson; karl.andersson@ltu.se

Received 8 August 2017; Accepted 9 August 2017; Published 2 October 2017

The International Telecommunications Union has provided vital statistics on ICT use, which evidently shows an increasing trend of universal growth in ICT uptake. According to ITU, in 2013, there are almost as many mobile-cellular subscriptions as people in the world. This is attributed to the mobile revolution, which delivers ICT applications in education, business, government, banking, health, and so on. Thus, the concept of mobile computing has revolutionized people's lives such that, nowadays, any information can be accessed and transmitted anytime and anywhere from any device. This new way of interacting with services basically has two main underlying drivers: (1) availability of innovative mobile devices (e.g., smartphones and tablets) and (2) emerging ubiquitous and affordable wireless networks. According to IndustryWeek, mobile computing is accelerating in organizations as well and mobile technology is viewed as providing the biggest boost to businesses. By 2020, the number of connected devices globally is expected to reach 80 billion and augmented reality facilitated via mobile devices will penetrate most sectors: leisure, social media, retail, restaurants, property, transport, and so on.

However, mobile computing is also a power-hungry technology. Therefore, there is a pressing need for the deployment of sustainable mobile computing and communication which focus on reduced energy consumption of its mobile ICT infrastructure and CO2 emissions in order to uphold the three pillars of sustainability: environmental, economic, and social. This is made possible if sustainable mobile computing platforms appropriately address the following challenges: extension of battery life, reduction of heat dissipation from components, power consumption, energy efficient wireless communications, efficient resource management, and sustainable power architecture. Some of the energy efficient techniques in sustainable mobile computing are as follows: (i) hardware: homogenous and heterogeneous multiprocessor system-on-chip, integration of multicore processors and multiple accelerometers into mobile system-on-chip, context-aware and energy aware operating systems, energy aware virtual memory (PAVM), application processor integration, and optimized scheduling in multiprocessor architectures; (ii) Wi-Fi communication: energy aware cellular data scheduling and energy-delay trade-off and dynamic voltage and frequency scaling (DVFS); (iii) sensors: accuracy and energy trade-off; (iv) applications: energy efficient computation off-loading, multithreaded applications, and green software design patterns.

The scope of this special issue is in line with recent contributions from academia and industry on the recent activities that tackle the technical challenges making mobile computing and communications more sustainable. For the current issue, we are pleased to introduce a collection of papers covering a range of topics such as frameworks for multicore mobile computing; architectures for IoT with mobile fog servers; energy efficient schemes for video streaming on mobile devices; security service chaining solutions for sustainable mobile edge computing; content downloading techniques for VANETs.

Acknowledgments

As always, we appreciate the high quality submissions from authors and the support of the community of reviewers.

Karl Andersson
Eric Rondeau
Ah-Lian Kor
Dan Johansson

Hindawi
Mobile Information Systems
Volume 2017, Article ID 4863167, 9 pages
https://doi.org/10.1155/2017/4863167

Research Article

Content Downloading with the Assistance of Roadside Cars for Vehicular Ad Hoc Networks

Haigang Gong[1] and Lingfei Yu[2]

[1]*School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China*
[2]*Hangzhou College of Commerce, Zhejiang Gongshang University, Hangzhou 310018, China*

Correspondence should be addressed to Lingfei Yu; linphie@163.com

Received 3 March 2017; Revised 4 July 2017; Accepted 30 July 2017; Published 11 September 2017

Academic Editor: Karl Andersson

Plenty of multimedia contents such as traffic images, music, and movies pose great challenges for content downloading due to the high mobility of vehicles and intermittent connectivity for vehicular ad hoc networks. Roadside units or APs can improve the efficiency of content downloading but with the cost of large investments. In this paper, an efficient content downloading scheme is proposed with the assistance of parking clusters, which are formed by roadside parked cars. After receiving the downloading request, the parking clusters, which the downloader will travel through according to the estimated trajectory, will make a download scheduling for the downloader. Then the downloader acquires the content chunks while it drives through the parking clusters. Simulation results show that the proposed scheme achieves better performance than intervehicle approach and RSU based approach.

1. Introduction

Vehicular ad hoc networks have been envisioned to be promising in many applications such as road safety and the intelligent transportation system (ITS). To facilitate road safety and enjoyable trip, there is a large amount of multimedia in the network, such as traffic image, surveillance video, music, and movie provided by the content provider. One of the most import requirements for VANETs is to distribute these multimedia-rich contents to the mobile vehicles.

However, content downloading in VANETs poses great challenges due to the short radio communication, dynamic topology, lack of bandwidth, intermittent connectivity, and high mobility of VANET. In the early days, a vehicle can request the content from the source directly with the Internet interface, but it achieves poor performance for the narrow bandwidth. With the development of broad bandwidth technologies such as WiMax and 4G, a vehicle has sufficient bandwidth to access the Internet, but with expensive accessing fee. In vehicle to vehicle (V2V) communication [1, 2], also called intervehicle communication, the content is transferred between the two vehicle nodes when they encounter. Intervehicle communication improves a little due to the highly intermittent connectivity, especially for larger contents. Obviously, infrastructures built in VANETs such as roadside units (RSUs) or Access Points (APs) could dramatically improve the performance of content downloading with high bandwidth. Some of works [3–5] employ WIFI-based APs for content distribution and perform better than the intervehicle communication based schemes. Unfortunately, infrastructure-based methods also have some drawbacks. Firstly, static roadside infrastructure is hardly adaptive to rapid-changing traffic. Secondly, the deployment of APs influences the performance heavily. The sparser the placement of APs, the worse the performance. However, Internet APs need costly installation of power and wired network connectivity, of which the costs can be as high as 5,000 US dollars per unit [6]. Generally, though infrastructure does improve connectivity, it often requires a large amount of investment and elaborate design, especially at the city scale.

Recent efforts [7–10] show that parked vehicles can play the role of the roadside unit. Since parking is a common phenomenon in most cities in our daily life, parked vehicles, especially on-street parked vehicles, are natural alternative of the roadside unit. With on-board wireless device and rechargeable battery, parked cars can communicate with any

cars driving through them. With the assistance of cars parked at the roadside, we propose an efficient content downloading scheme for vehicular ad hoc networks. The roadside parked cars are grouped into parking clusters on each road. When a vehicle, called downloader, submits a downloading request to the parking cluster, the parking cluster will know the following trajectory of the downloader according to a trip history model. Then it will notify the parking clusters, which the downloader will drive through, to download a part of the content from the content provider. While the downloader travels through the parking clusters, it will get different parts of the content from the parking clusters.

The remainder of this paper is organized as follows. Section 2 discusses the related works. The problems are described in Section 3. In Section 4, the proposed scheme is presented in detail. Section 5 introduces simulations and discusses the results. The last section concludes the paper.

2. Related Works

There have been lots of research works on content sharing and delivery in mobile ad hoc networks [11–13]. Particularly, content downloading is also a hot issue in vehicular ad hoc networks. Ota et al. [14] investigate cooperative content downloading in the scenario of highway VANET. A P2P-like scheme [2] enables vehicles to exchange small content chunks when they encounter, but with poor performance.

To improve the performance, some techniques are introduced. Li et al. [15] propose CodeOn, employing symbol level network coding (SLNC) to combat the lossy wireless transmissions, which is robust to transmission errors and encourages more aggressive concurrent transmissions. In [16], Wang et al. propose a cooperative approach based on coalition formation games, in which OBUs exchange their possessed pieces by broadcasting to and receiving from their neighbors. Though they achieve better performance, this depends on the performance of OBUs. In [17], Malandrino et al. outline the performance limits of such a vehicular content downloading system by modeling the downloading process as an optimization problem and maximizing the overall system throughput.

Naturally, roadside infrastructure [3–5] is employed for content distribution in vehicular networks. Vehicular users can download large files from servers in the Internet through roadside Access Points (APs). Considering that vehicular content downloading via open WiFi Access Points (APs) can be challenging due to sparse AP deployment with bounded communication range and the rapid movement of traveling vehicles, Chen et al. [18] discuss joint resource allocation and scheduling problem for efficient content downloading considering channel contention and scarce AP resource utilized effectively. To get over that drawback and to improve a collaborative downloading, a P2P network [19] is constructed among the OBUs which fall out of the RSUs coverage and a new cell-based clustering scheme is proposed, which organizes the RSUs into a cluster, so as to improve delivery efficiency.

In these schemes, server periodically delivers the pieces of data to other vehicles, and then vehicles that obtain chunks transport the data according to a carry-and-forward paradigm to the destination vehicle. However, the highly dynamic network topology introduces the intermittent connectivity, which causes unsuccessful downloading.

Compared with the high cost of infrastructure, parked cars on the roadside can be leveraged to play the role of the RSUs. Liu et al. [6] present the idea of PVA (Parked Vehicle Assistance) firstly. They investigate that parked vehicles are natural alternative for roadside units and do not need any deployment investment.

In [7], Malandrino et al. present a content downloading system in vehicular networks using parked vehicles. The goal is to share big pieces of data between vehicles and maximize content freshness and utilize the radio resources. But data exchange only involves one-hop communication. Moreover, they investigate the possibility of exploiting parked vehicles to extend the RSU service coverage. It leverages optimization models aiming at maximizing the freshness of content that downloaders retrieve, the efficiency in the utilization of radio resources, and the fairness in exploiting the energy resources of parked vehicles [20].

Liu et al. propose ParkCast [8] in their following studies, which employs roadside parked vehicles to deliver content in urban VANETs. However, ParkCast does not provide a specific strategy to be tackled with the unfinished downloading, especially for the large size of content.

In [9], an energy efficient data dissemination protocol with roadside parked cars' assistance in VANETs is proposed. Similarly to ParkCast, the parked cars on the roadside are organized into several clusters, which store and distribute the media files for moving vehicles.

Noticeably, the works on content downloading leveraging parked vehicles all assume that the content has been already stored in the parked cars, which is impractical.

3. Problem Statement

With the support of RSUs, vehicles can request content from RSUs. As in Figure 1, vehicle A enters into the road and sends a downloading request to RSU_1, which will get the content from the content provider, and then RSU_1 transmits the content to vehicle A. If the content is too large, vehicle A will request the remainder of the content from other RSUs such as RSU_2. As mentioned before, the high cost of RSUs deployments poses a heavy burden for the city government. Moreover, the optimal deployment of RSUs is also a challengeable task due to budget limits, street layout, and traffic changes. For example, if vehicle A drives into a road without RSUs, it will not get the remainder of the content until it encounters another RSU deployed in other road, leading to long transmission delay and poor experiences.

Fortunately, the parked cars on the roadside could be a natural alternative of RSUs. A survey [11] explores on-street parking in Ann Arbor and the US state of Michigan. It found that the utilization of the parking spaces is quite stable, although each parking is short and undulated. Occupancy ratio averages 93.0% one day during the peak. Even off-peak occupancy ratio averages almost 80%. For all practical purposes, the on-street parking spaces are used all of the

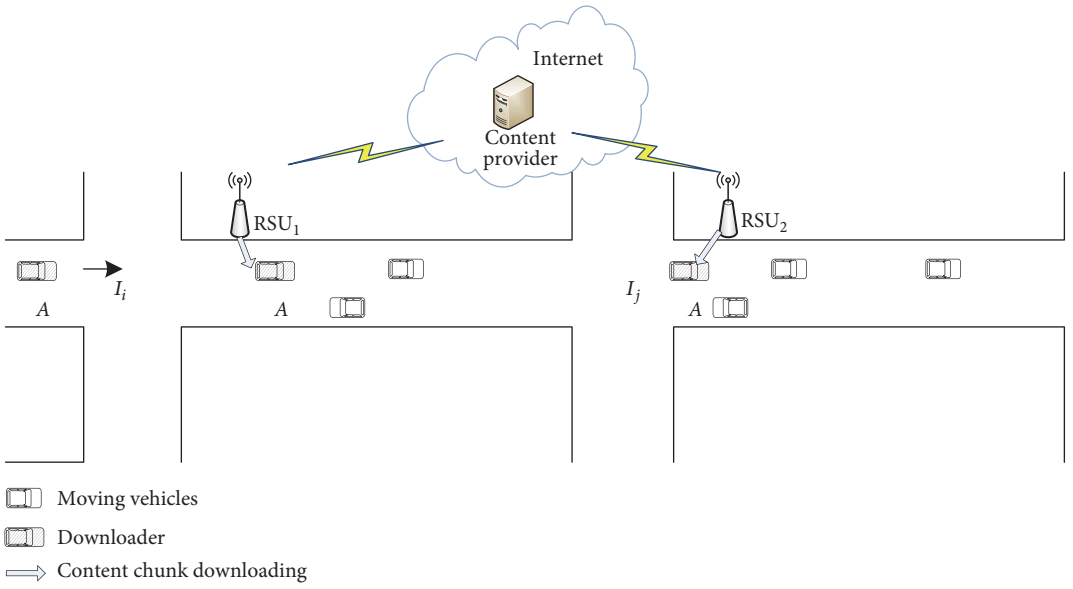


Figure 1: Content downloading via RSUs.

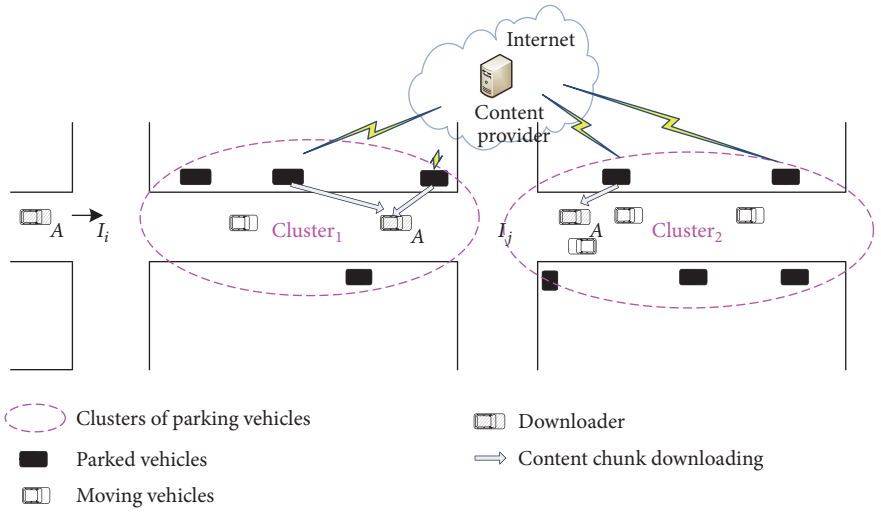


Figure 2: Content downloading via roadside parked cars.

time, for high parking demand. Consequently, the roadside parking spaces can be thought to be frequently occupied and parking lots have some "fixed" cars, as stable roadside units in communication.

The roadside parked cars can be organized into clusters and play the role of RSUs. As in Figure 2, vehicle A enters into the road; it submits a content downloading request to Cluster_1. The cluster members will download the content from the Internet and transmit the content to vehicle A. In the meantime, Cluster_1 will notify Cluster_2, into which vehicle A will drive, to download the remainder of the content, so that vehicle A can get the remainder of the content quickly.

There are some assumptions before we present the details of parked cars based content downloading scheme. Firstly, we assume that vehicles are equipped with GPS and electric maps, which are of low cost and popular nowadays. Based on GPS and electric maps, vehicles can get current speed, location, and moving direction. Then the trajectory of the vehicles can be recorded and saved. It is also a basic assumption for the intervehicle scheme because the cars need to know their location, by which the relay node is decided when they encounter. For RSU based approach, the GPS information can be got from the roadside unit. Secondly, we assume that the On-Board Units (OBUs) on vehicles are powered by the car battery, which could be charged while driving, for supporting the communication in parking. In [21], it is demonstrated that the energy of the car battery can power the OBUs for 80 hours or less. It is not necessary to worry about the depletion of the car battery because the average duration of street parking only lasts 6.64 hours, according to a real urban parking report [22]. Finally, we assume that some owners of the parked cars are willing to share the resources during parking.

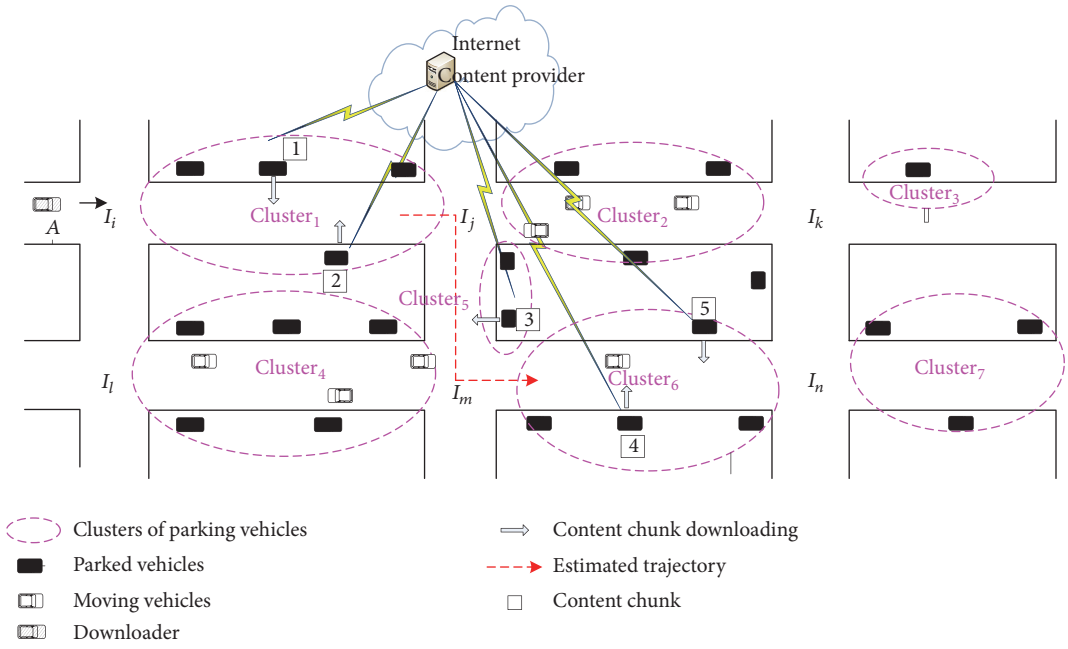


Figure 3: System overview.

It can be motivated by some incentive mechanisms. Actually, according to the experience of P2P file systems [23], there are still 30% users who are willing to share their resources, even if there are not any incentives. That is to say, even though they cannot benefit from the sharing, some users are cooperative to contribute resources.

4. The Detailed Design

4.1. System Overview. Figure 3 shows the overview of the network. In Figure 3, there are 6 intersections (I_i, I_j, I_k, I_l, I_m, and I_n) and several road segments in the map. Black vehicles are parked at the roadside and white vehicles move along the road. Vehicle A wants to download the content from the content provider. Firstly, the parked cars on the roadside are grouped into clusters. Downloader A sends its request for a specific content to Cluster$_1$. Cluster$_1$ estimates A's following trajectory according to a trip history model and knows which clusters the downloader will drive through. Considering the low bandwidth of the network, the content may not be downloaded while driving through the road segment. Assume that the content is divided into several chunks. Cluster$_1$ will make a schedule to tell downloader A which chunks can be acquired from which clusters. At the same time, Cluster$_1$ notifies other clusters, which the downloader will travel through, to download the specified chunks in advance, so that the chunks could be transmitted to the downloader once it enters into these clusters.

4.2. Clustering of Parked Cars. Due to the high stability and utilization of roadside parking, clustering parked vehicles on roadside is feasible in the city [24]. All parked vehicles (PV) on one road are organized into a virtual cluster, even if some of them cannot communicate with each other directly. This is viable, for the moving vehicles (MV) will travel across the road and help to maintain the connectivity of the whole virtual cluster. For supporting content downloading, the cluster head of virtual cluster needs to handle the following three tasks: (1) The first of them is cluster management, including membership management. (2) The second is content download scheduling. It decides the downloader that gets content chunks from each cluster member. If the downloader cannot finish downloading on the current road, it notifies other clusters that the mobile car will drive through to download the rest of content chunks. (3) The third is some information distributed over all the clusters, such as the average travel time on each road, which can be used to estimate how much chunks can be acquired on the road. As cluster members, there are three roles: (1) downloading the chunks from the content provider based on the scheduling of the cluster head; (2) distributing the chunks over the cluster, so that it can be a natural database to store the content; (3) transmitting the chunks to the downloading cars.

The election of cluster head (CH) is simple. The parked car closest to the intersection could be the cluster head. As in Figure 4, there are 3 scenarios of parked vehicles clustering. If l_{ij} is less than the communication radius of the vehicles, R, just one cluster head is needed to manage the cluster, as shown in Figure 4(a). In Figure 4(b), if l_{ij} is greater than R but it is less than $2R$, there may be two cluster heads (i.e., CH$_i$ and CH$_j$) at the two ends of road r_{ij}, so that moving vehicles could get the information of the virtual cluster as soon as possible once they enter into the road from I_i or I_j. When l_{ij} is greater than $2R$, there might be some isolated parked vehicle (IPV) that cannot communicate with any cluster members directly. As in Figure 4(c), the moving vehicle MV$_1$ would help the IPV communicate with the CH$_i$ or CH$_j$ and maintain the virtual cluster.

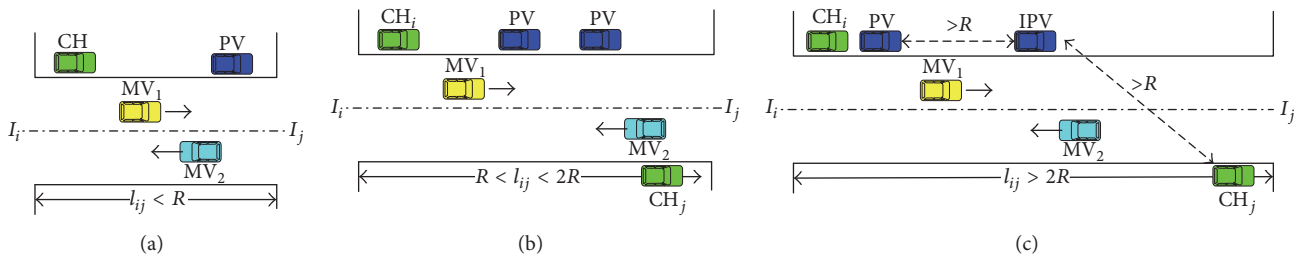


Figure 4: Different scenario of clustering.

Table 1: Trip history records.

Week	Day	Time	Source	Route	Destination
1	Fri.	<10	Home	$r_{12} \to r_{23} \to r_{35} \to r_{57} \to \cdots \to r_{79}$	Work place
1	Fri.	16–18	Work place	$r_{79} \to r_{57} \to r_{35} \to r_{23} \to r_{12}$	Home
1	Sat.	14–16	Home	$r_{12} \to r_{24} \to \cdots$	Sports
1	Sat.	>18	Sports	$\cdots$	Friend's home

4.3. Trajectory Prediction. We use the trip history mobility model presented in our previous work [25] to predict the following trajectory of the vehicle. The idea of the mobility model is to record the trip details of vehicles every day, as shown in Table 1. The start time of the trip is separated into discrete time sets: Day (Mon., Tues., Wed., Thurs., Fri., Sat., and Sun.) and Time (<10, 10–12, 12–14, 14–16, 16–18, >18). Once a user starts his car, the start time and location are recorded as "Day," "Time," and "Source." While moving, the vehicle gets position data from the GPS every several seconds. When the car stops, the last position is denoted by "Destination." Then a decision tree can be constructed based on the trip history records to predict vehicle's moving trajectory.

Figure 5 depicts the decision tree according to Table 1, where the history trip records are expressed as branches and leaves. The probability of selecting a destination is given by

$$P_{kq} = \frac{f_{kq}}{N_k}, \tag{1}$$

where N_k is the total number of trips starting at the root node k, and f_{kq} is the number of trips according to specific time and destination choices at the leaf node kq. P_{kq} represents the probability of moving toward the destination in history. Since the vehicle knows current Source, Day, and Time, it can find a Destination choice with a maximum P_{kq} in the tree as an initial prediction. While driving, the vehicle periodically checks whether its location is on the way to the predicted destination. If not, the vehicle needs to calculate a new destination probability by

$$P_{kq} = \begin{cases} 0, & \text{if route is impossible,} \\ \dfrac{f_{kq}}{\sum f_k}, & \text{otherwise,} \end{cases} \tag{2}$$

where f_k is the total number of the rest of the trips after removing the infeasible ones. Then it can find a new destination prediction with maximum P_{kq}. According to the model, each vehicle can predict one's moving trajectory. If it wants to download content, it will notify the cluster head about the predicted trajectory, which can be utilized to make a downloading schedule.

4.4. Content Downloading. A content file, assigned by a content identification (CID), is cropped into several chunks, each of which is also assigned a chunk identification (ChunkID). When downloader A wants to download a content on the road r_{ij}, it will send the cluster head of the road r_{ij} a request, containing a 4-tuple composed of CID, $\text{Loc}_A\text{Vec}_A T_A$, and Route_A, where Loc_A and Vec_A are the current location and the velocity of vehicle A, T_A is a vector of average travel time on each road in the following trajectory, and Route_A is the following trajectory predicted by vehicle A, according to the trip history model. The cluster head will know which clusters downloader A will bypass and can compute the number of the chunks that the downloader can download on each road in the following trip, as in

$$N_{ij} = \left\lfloor \frac{\int_0^{t_{ij}} C \cdot dt}{S} \right\rfloor, \tag{3}$$

where C is the MAC throughput that can be estimated in [26], and S is the amount of a chunk. The cluster head will notify downloader A about the downloading schedule that specifies which clusters and how much chunks can be downloaded on each road r_{ij} in the predicted trajectory. Meanwhile, the cluster head also notified other cluster heads on the following trajectory to download the specified chunks from the content provider.

As shown in Figure 3, downloader A calculates its following trajectory and sends the 4-tuple request to the cluster head of Cluster_1. Based on (3) and 4-tuple, the cluster head knows the content is composed of 5 content chunks and makes a schedule where the first 2 chunks can be acquired from Cluster_1, the third chunks can be got from Cluster_5, and the last 2 chunks can be downloaded from Cluster_6. The schedule

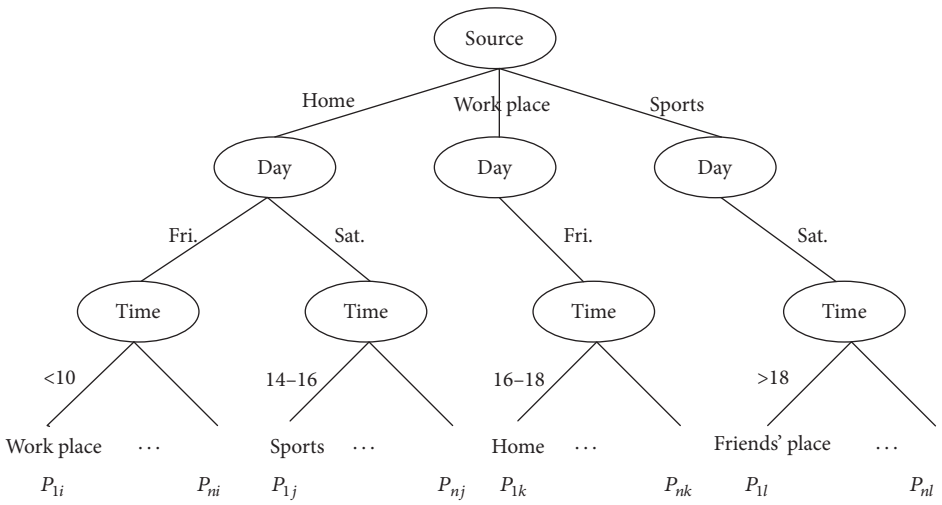


Figure 5: Structure of decision tree.

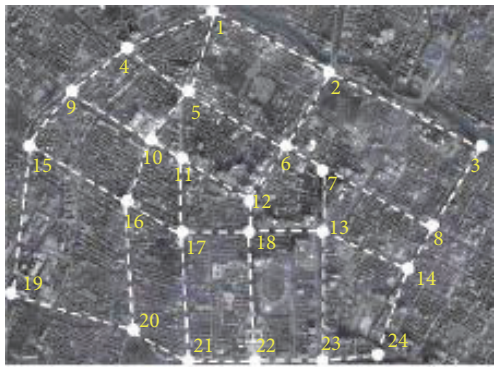


Figure 6: Road topology in survey and simulation.

Table 2: Simulation parameters.

Parameters	Value
Number of vehicles	50~300
Velocity (km/h)	40~60
MAC	802.11
Data rate (bps)	2 M
Radio range (m)	250
Size of content (Mb)	5~100
Size of chunk (Mb)	2.5
Ratio of downloader	5% ~25%
Transmitting power, P_T (W)	1.5
Receiving power, P_R (W)	0.8
Idling power, P_I (W)	0.2
Computing power, P_C (W)	4.0
Power of the car battery (W)	960

will be transferred to the downloader and the cluster heads of Cluster_5 and Cluster_6. Moreover, the cluster heads will choose some cluster members to download the specified chunks from the content provider. When the downloader bypasses these cluster members, the chunks will be transmitted to the downloader.

5. Evaluation

5.1. Simulation Setup. Firstly, we perform weeks' survey on an urban area of Chengdu, a city in China. As shown in Figure 6, a real street map with the range of 3600 m∗2500 m is selected, including 24 intersections and 35 bidirectional roads. Each intersection is marked by a number from 1 to 24. During the survey, we investigated the traffic and roadside parking statistics at different time during a day.

There are three kinds of streets with different parking limits. The first kind permits free parking at roadside, as $R_{1,2}$, $R_{2,3}$, and so on, which results in a very high node density as 300 veh/km in average. The second one, as $R_{1,4}$ and $R_{4,9}$, lacks public parking spaces. These streets have a very low vehicle density as 20 veh/km, which comes from some reserved parking spaces and illegal parking. The third one has a moderate vehicle density as 98 veh/km.

In simulation, VanetMobiSim-1.1 is used to generate realistic urban mobility traces, which can be directly utilized by the network simulator NS-2.33. The history trajectory records are collected from 30 volunteers who travel through the area of the map in Figure 6 for 4 weeks. And each downloader will be assigned a history trajectory record, which is chosen from the 30 history trajectory records randomly. For evaluating the energy consumption, the power model in [27] is used, including the transmitting power, receiving power, and idling power, listed in Table 2. Moreover, the computing power of the OBUs and the power of the car battery are 4.0 W and 960 W, given by [21], respectively. In the simulation, parked vehicles are located on random positions of each street, following the densities collected in survey. Other simulation parameters are shown in Table 2.

We evaluate our scheme with intervehicle approach [2] and RSU based approach [3]. The downloaders are chosen from the moving cars randomly, and each of downloaders

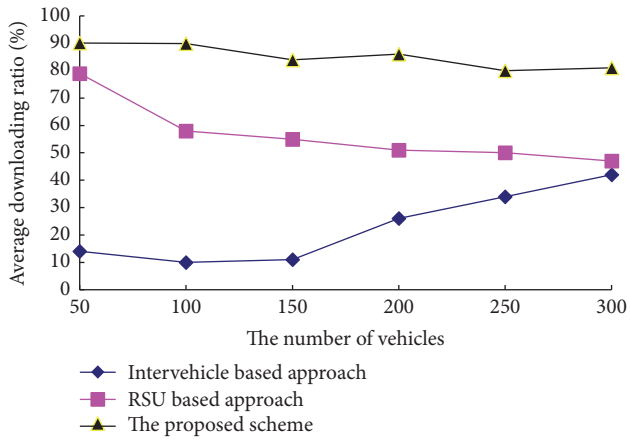


Figure 7: Impact of the number of mobile vehicles.

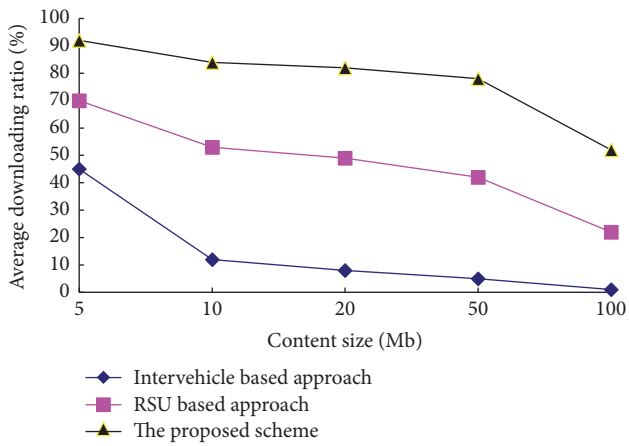


Figure 8: Impact of the size of content.

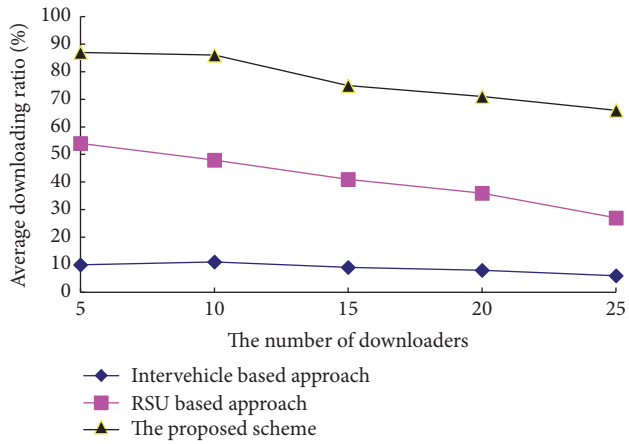


Figure 9: Impact of the number of downloaders.

will be assigned a history trajectory record, according to volunteers' travels. For RSU based approach, the number of RSUs is crucial for the performance. As in [3], the number of RSUs is set to 5, and the content files are stored in these 5 RSUs. When the downloader travels through the RSU, it can submit a content request to the RSU, which will download the content from the content provider and then transmit to the downloader. For the intervehicle approach, the downloader and its neighbors are cooperative to download the content file from the content provider. For our proposed scheme, the content file will be pieced into several chunks, and cluster members will download chunks for the downloader and transmit them to the downloader when the downloader bypasses the cluster members.

5.2. Simulation Results. Comparing the protocols, we choose to evaluate them according to the average downloading ratio, defined as the ratio between the number of requests that successfully download the content and the total amount of requests.

We firstly evaluate the average download ratio of the three approaches under different number of mobile vehicles and the results are shown in Figure 7. The size of content is set to 10 Mb. Moreover, the percentage of the downloaders that request content is set to 5%.

As shown in Figure 7, for the intervehicle scheme, the more the mobile vehicles are, the more the chances to maintain connection with neighbors are, so that the downloading ratio increases rapidly. Though the average downloading ratios of the RSU based method and our proposed scheme decrease while increasing the number of mobile vehicles, the intervehicle performs poorer than that of the two approaches due to intermittent communication. Our proposed scheme performs much better than the RSU based approach, reaching 90% or so, compared with 55% of the RSU based approach when there are 150 mobile vehicles. The reason is that the number of RSUs is not enough to cover all roads, compared with the roadside parked cars almost covering each road. The more mobile vehicles means more downloaders, leading to unsuccessful downloading due to collision.

Figure 8 describes the impact of the content size on the average downloading ratio, while there are 150 mobile vehicles and 10 vehicles request to download the content. As in Figure 8, the average downloading ratio decreases with the increasing of the content size of all schemes quickly. For our proposed scheme, the reason is that large size of the content means that there are more chunks for the content. It needs more parking clusters to participate in transferring the chunks, which is not satisfied in the simulation map. Similarly, the RSU based approach has no enough RSUs and high bandwidth to support the transmission of large size content. However, the download ratio of our proposed scheme has still 65% much more than that of the RSU based approach. For the intervehicle approach, large size of the content file means it needs more neighbors to download the content cooperatively, which is hard to be satisfied if there are not enough mobile vehicles, leading the rapid decreasing of the downloading ratio.

Figure 9 plots how the ratio of downloader influences the performance of all schemes with 10 M content size and 150 mobile vehicles. In Figure 9, the average downloading ratio of all schemes decreases when the number of downloaders increases. Due to the resources of roadside parking clusters,

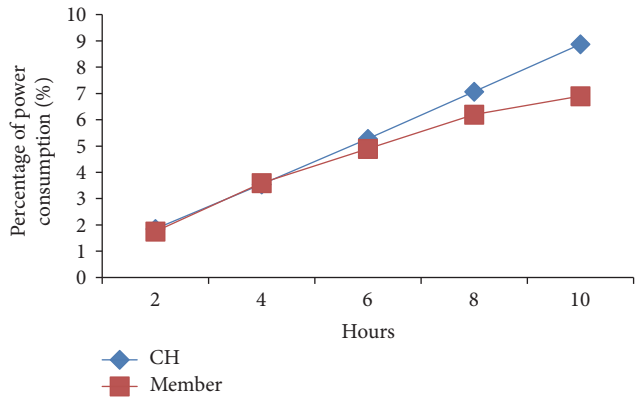


Figure 10: The power consumption of cluster head and cluster member.

our proposed scheme achieves the highest downloading ratio, compared with the poorest performance of intervehicle based approach. Of course, the intervehicle approach performs poorest. With the increase of downloaders, they need more neighbors to collaborate to download the content. However, it is not an easy task and will introduce more collision.

To study the potential effect on power consumption of the parked cars, the downloader will initiate a new request for another content file periodically, until the end of simulation. The energy consumption is calculated by

$$E = P_T t_T + P_R t_R + P_I t_I + P_C t_C, \quad (4)$$

where t_T, t_R, and t_I are the time that the transceiver stays in the status of transmitting, receiving, and idling. And t_C is the time spent on computation of the OBUs. The results have been shown in Figure 10. Obviously, the power consumption of both cluster head and cluster member increases with time elapsing. In the earlier stage of simulation, the cluster head consumes a little more energy than the cluster member. But in the latter, the cluster head consumes more energy than the cluster member. The reason is that chunks are distributed over the networks by all parked cars including the cluster head and the cluster member in the beginning. Once the chunks are stored in all parked cars, the cluster member does not distribute the chunks and only transmits the chunks to the downloader. Though the power consumption of the cluster head is more than that of the cluster member, the percentage of power consumption of the cluster head is just 8.8% of the car battery after 10 hours of parking. According to [22], the average duration of street parking only lasts 6.64 hours, meaning that the car battery can sufficiently supply the energy for communication while parking. Practically, a threshold can be set to ensure that energy of the battery can fire the engine of the car.

6. Conclusion

In this paper, a content downloading scheme with the assistance of roadside parked cars is proposed. The parked cars, which form a virtual cluster, play the role of RSUs, downloading the content from the content provider and transmitting it to the downloader. Clusters make a schedule to notify the downloader about how to acquire content chunks from which clusters based on the estimated trajectory of the downloader. Simulation results show that our proposed scheme achieves 65% higher average download ratio than RSU based approach.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is supported in part by National Science Foundation of China under Grants nos. 61370204 and 61572113 and Zhejiang Provincial Natural Science Foundation under Grant no. LQ16F02001.

References

[1] I. Leontiadis, P. Costa, and C. Mascolo, "A hybrid approach for content-based publish/subscribe in vehicular networks," *Pervasive and Mobile Computing*, vol. 5, no. 6, pp. 697–713, 2009.

[2] W. Zhao, Y. Qin, and Y. Cheng, "An efficient downloading service of large popular files in vanet based on 802.11p protocol," *International Journal of Distributed Sensor Networks*, vol. 2015, Article ID 824294, 2015.

[3] Y. Zhang, J. Zhao, and G. Cao, "Service scheduling of vehicle-roadside data access," *Mobile Networks and Applications*, vol. 15, no. 1, pp. 83–96, 2010.

[4] P. Deshpande, A. Kashyap, C. Sung, and S. R. Das, "Predictive methods for improved vehicular WiFi access," in *Proceedings of the 7th ACM International Conference on Mobile Systems, Applications, and Services, MobiSys'09*, pp. 263–276, June 2009.

[5] Jupiter research, "Municipal wireless: partner to spread risks and costs while maximizing benefit opportunities," Tech. Rep., Jupitermedia Corporation, 2005.

[6] N. Liu, M. Liu, W. Lou, G. Chen, and J. Cao, "PVA in VANETs: stopped cars are not silent," in *Proceedings of the IEEE INFOCOM 2011*, pp. 431–435, April 2011.

[7] F. Malandrino, C. Casetti, C. F. Chiasserini, C. Sommer, and F. Dressler, "Content downloading in vehicular networks: bringing parked cars into the picture," in *Proceedings of the 2012 IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications, (PIMRC '12)*, pp. 1534–1539, September 2012.

[8] N. Liu, M. Liu, G. Chen, and J. Cao, "The sharing at roadside: vehicular content distribution using parked vehicles," in *Proceedings of the IEEE Conference on Computer Communications, (INFOCOM '12)*, pp. 2641–2645, March 2012.

[9] H. Gong, L. Yu, and X. Zhang, "Social contribution-based routing protocol for vehicular network with selfish nodes," *International Journal of Distributed Sensor Networks*, vol. 2014, Article ID 753024, 12 pages, 2014.

[10] A. Nandan, S. Das, G. Pau, M. Gerla, and M. Y. Sanadidi, "Cooperative downloading in vehicular ad-hoc wireless networks," in *Proceedings of the 2nd Annual International Conference on Wireless On-Demand Network Systems and Services (WONS '05)*, pp. 32–41, January 2005.

[11] Y. Li, Z. Wang, D. Jin, and S. Chen, "Optimal mobile content downloading in device-to-device communication underlaying cellular networks," *IEEE Transactions on Wireless Communications*, vol. 13, no. 7, pp. 3596–3608, 2014.

[12] H. Li, Y. Yang, X. Qiu, Z. Gao, and G. Ma, "Cooperative downloading in mobile ad hoc networks: a cost-energy perspective," *International Journal of Distributed Sensor Networks*, vol. 2016, Article ID 3028642, 2016.

[13] H. Gong, L. Yu, and X. Zhang, "Social contribution-based routing protocol for vehicular network with selfish nodes," *International Journal of Distributed Sensor Networks*, vol. 2014, Article ID 753024, 12 pages, 2014.

[14] K. Ota, M. Dong, S. Chang, and H. Zhu, "MMCD: cooperative downloading for highway VANETs," *IEEE Transactions on Emerging Topics in Computing*, vol. 3, no. 1, pp. 34–43, 2015.

[15] M. Li, Z. Yang, and W. Lou, "CodeOn: cooperative popular content distribution for vehicular networks using symbol level network coding," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 1, pp. 223–235, 2011.

[16] T. Wang, L. Song, Z. Han, and B. Jiao, "Dynamic popular content distribution in vehicular networks using coalition formation games," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 9, pp. 538–547, 2013.

[17] F. Malandrino, C. Casetti, C.-F. Chiasserini, and M. Fiore, "Optimal content downloading in vehicular networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 7, pp. 1377–1391, 2013.

[18] L. Chen, Z.-J. Li, and S.-X. Jiang, "Content downloading-oriented resource allocation joint scheduling in drive-thru networks," *Ruan Jian Xue Bao/Journal of Software*, vol. 25, no. 10, pp. 2362–2372, 2014.

[19] W. Huang and L. Wang, "ECDS: Efficient collaborative downloading scheme for popular content distribution in urban vehicular networks," *Computer Networks*, vol. 101, pp. 90–103, 2016.

[20] F. Malandrino, C. Casetti, C.-F. Chiasserini, C. Sommer, and F. Dressler, "The role of parked cars in content downloading for vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 9, pp. 4606–4617, 2014.

[21] J. Balen, G. Martinovic, K. Paridel, and Y. Berbers, "PVCM: assisting multi-hop communication in vehicular networks using parked vehicles," in *Proceedings of the 2012 4th International Congress on Ultra Modern Telecommunications and Control Systems, (ICUMT '12)*, pp. 119–122, October 2012.

[22] C. Morency and M. Trépanier, "Characterizing parking spaces using travel survey data," *CIRRELT*, vol. 10, no. 3, pp. 25–33, 2006.

[23] S. Saroiu, P. K. Gummadi, S. D. Gribble, M. G. Kienzle, and P. J. Shenoy, "A measurement study of peer-to-peer file sharing systems," in *Proceedings of the Electronic Imaging 2002*, pp. 156–170, San Jose, CA, USA.

[24] A. Adiv and W. Wang, "On-street parking meter behavior," *Journal of Transportation Quarterly*, vol. 41, no. 1, pp. 281–307, 1987.

[25] H. Gong, L. Yu, N. Liu, and X. Zhang, "Mobile content distribution with vehicular cloud in urban VANETs," *China Communications*, vol. 13, no. 8, Article ID 7563691, pp. 84–96, 2016.

[26] T. H. Luan, X. Shen, and F. Bai, "Integrity-oriented content transmission in highway vehicular ad hoc networks," in *Proceedings of the 32nd IEEE Conference on Computer Communications (INFOCOM '13)*, pp. 2562–2570, IEEE Press, April 2013.

[27] P. Serrano, M. Hollick, and A. Banchs, "On the trade-off between throughput maximization and energy consumption minimization in IEEE 802.11 WLANs," *Journal of Communications and Networks*, vol. 12, no. 2, pp. 150–157, 2010.

Hindawi
Mobile Information Systems
Volume 2017, Article ID 9642958, 11 pages
https://doi.org/10.1155/2017/9642958

Research Article

An Adaptive and Integrated Low-Power Framework for Multicore Mobile Computing

Jongmoo Choi,[1] Bumjong Jung,[1] Yongjae Choi,[1] and Seiil Son[2]

[1]*Department of Software, Dankook University, Yongin, Republic of Korea*
[2]*Korea Communications Agency, Daejeon, Republic of Korea*

Correspondence should be addressed to Jongmoo Choi; choijm@dankook.ac.kr

Received 20 January 2017; Accepted 15 March 2017; Published 12 June 2017

Academic Editor: Karl Andersson

Employing multicore in mobile computing such as smartphone and IoT (Internet of Things) device is a double-edged sword. It provides ample computing capabilities required in recent intelligent mobile services including voice recognition, image processing, big data analysis, and deep learning. However, it requires a great deal of power consumption, which causes creating a thermal hot spot and putting pressure on the energy resource in a mobile device. In this paper, we propose a novel framework that integrates two well-known low-power techniques, DPM (Dynamic Power Management) and DVFS (Dynamic Voltage and Frequency Scaling) for energy efficiency in multicore mobile systems. The key feature of the proposed framework is adaptability. By monitoring the online resource usage such as CPU utilization and power consumption, the framework can orchestrate diverse DPM and DVFS policies according to workload characteristics. Real implementation based experiments using three mobile devices have shown that it can reduce the power consumption ranging from 22% to 79%, while affecting negligibly the performance of workloads.

1. Introduction

Intelligent services are actively introduced into mobile computing environments [1, 2]. For instance, modern black box devices support not only the traditional recording service but also ADAS (Advanced Driver Assistance Systems) such as blind spot monitoring, pedestrian detection, and automatic emergency braking. Smartphones provide voice recognition and image processing for better HCI (Human Computer Interface) and big data processing and deep learning for context-aware services with the consideration of user personality.

To support such intelligent services efficiently, mobile computing devices are equipped with multiple cores. Smartphones have the heterogeneous multicore architecture, called big.LITTLE, which consists of performance-optimized big cores and energy-optimized little cores with a single ISA (Instruction Set Architecture) [3]. Recently released embedded boards for IoT (Internet of Things) such as Raspberry Pi, Odroid, Edison, Jetson, and Artik also provide multiple cores [4–6].

However, employing multicore in mobile devices triggers a new issue. As the number of cores increases, the power consumption used by cores becomes a significant portion in mobile devices. The increased power consumption puts pressure on the energy resource in a mobile device due to the limitation in battery capacity [7]. In addition, it causes the high internal temperature in a mobile device [8], having a potential to result in the thermal runaway [9].

To address the multicore power consumption issue, two well-known techniques are devised [7, 10]. One is DPM (Dynamic Power Management), also known as *offlining*, which turns off individual cores in order to reduce the per-core power consumption [11]. The other technique is DVFS (Dynamic Voltage and Frequency Scaling), which decreases the frequency and voltage of a core. [12].

One interesting observation is that most mobile applications have limited parallelism [13–15]. For instance, Seo et al. analyze how the multiple cores are utilized in mobile devices using TLP (Thread Level Parallelism) and observe that TLP of most mobile applications is ranging from 1.4 to 3.9 [15]. It implies that the required active cores for the applications are

less than 3.9 on average, disclosing the opportunity of DPM and DVFS.

In this paper, we propose a new low-power framework for multicore mobile devices. It supports both DPM and DVFS in an integrated manner. Also, it keeps track of the CPU load of applications and turns on/off cores or changes frequency adaptively so that it can reduce the power consumption while trying to minimize its impact on performance.

The framework consists of three components, namely, online resource usage monitor, lower-power controller, and policy manager. The online resource monitor gathers resource usage statistics such as CPU utilization and power measurement. The lower-power controller materializes the DPM and DVFS techniques using the Linux *CPU hotplug* and *governor* mechanism, respectively [16, 17].

Finally, the policy manager provides a rule regarding how to integrate DPM and DVFS techniques based on their overhead and effect on the power consumption. Also, it applies the integrated solution appropriately according to the resource usage characteristics of applications. In addition, it supports user interfaces so that a user can configure his/her own control parameters.

We have implemented the framework in the Linux kernel version 3.10 and have evaluated its effectiveness using three mobile devices. Experimental results show that the workload-aware adaptability of the framework can reduce the power consumption ranging from 22% to 79%, while not affecting the performance of workloads greatly. Also, we observe that the power reduction depends on the features of a mobile device, especially in the big.LITTLE architecture.

The rest of this paper is organized as follows. In Section 2, we explain previous studies related to this work. Then, our proposed framework is discussed in Section 3. Section 4 presents real implementation based experimental results. Finally, we summarize conclusion and future work in Section 5.

2. Related Work

As the energy efficiency becomes a critical issue, a lot of studies have been conducted for analyzing and enhancing the power consumption in mobile computing systems. In this section, we classify these studies into the following four categories: power consumption analysis, workload analysis, DPM/DVFS techniques, and system/application-level approach.

2.1. Power Consumption Analysis. Understanding of where and how power is consumed in mobile devices is a key requirement for efficient power management. Carroll and Heiser present a detailed analysis of power consumption in mobile phones using Google Nexus One, HTC Dream, and OpenMoko Neo Freerunner [18]. They provide not only the overall system power but also the breakdown of power consumed by the main hardware components. Their analysis shows that wireless communication and display are the heaviest power consumers. Also, CPU is identified as a heavy consumer especially for the CPU intensive workloads and in the suspended state.

As the number of cores increases, the power consumption used by cores also increases sharply. Several studies demonstrate that the power consumption increases with the number of active cores and as the voltage and frequency increase [6–8]. Tawara et al. also present how the power consumption affects the internal temperature using thermography [8]. Zhu and Shen observe that even though the power consumption increases with the number of cores, the first activated core incurs much higher power cost than each additional core does in the same processor due to the shared resources [6].

2.2. Workload Analysis. Even though employing multicore in mobile devices increases the power consumption, it also gives an opportunity to reduce the application execution time, eventually leading to energy saving. To explore the opportunity, several studies have examined how much parallelism is there in mobile applications [13–15].

Gao et al. analyze how the multiple cores are utilized in mobile devices using TLP (Thread Level Parallelism) [13]. They report that TLP of the most mobile applications including browser, map, music, and games is less than 2 on average, meaning that the number of active cores used by the majority of applications is below 2. Similar behavior is also observed by Zhang et al. [14]. Seo et al. investigate TLP using various mobile benchmark applications in mobile devices and observe that applications have TLP ranging from 1.4 to 3.9 [15]. These studies uncover the necessity to turn off cores or to adjust voltage and frequency adaptively according to different program execution phases.

2.3. DPM/DVFS Techniques. DPM and DVFS are two well-known techniques for dynamic energy-aware CPU managements. For DPM, Linux provides the CPU hotplug mechanism that allows cores to be added to or removed from a running kernel [16]. For DVFS, Linux supports the governor mechanism that permits user to adjust the CPU frequency [17]. There are several governors such as ondemand, powersave, and performance, which will be discussed further in Section 3.

Carroll and Heiser explore how to use DPM and DVFS to reduce power consumption [7, 19]. They propose a framework, called *medusa*, an offline-aware frequency governor, which integrates core offlining with frequency scaling. In addition, they find that modern smartphones have quite different characteristics, implying that policies that work well on one processor can lead to poor results on another. Tawara et al. design a framework, called *idle reduction*, which turns on/off cores or changes the CPU frequency dynamically according to the intensity of workloads [8].

These two works are closely related to our work in that they integrate DPM and DVFS and apply them adaptively based on workload characteristics. However, our proposed framework differs from them in the following three aspects: (1) we consider not only the homogeneous but also heterogeneous multicore devices, (2) we carefully separate policy and mechanism to support flexibility, and (3) we provide a configuration file with various control parameters so that a user can easily configure his/her preferred policy.

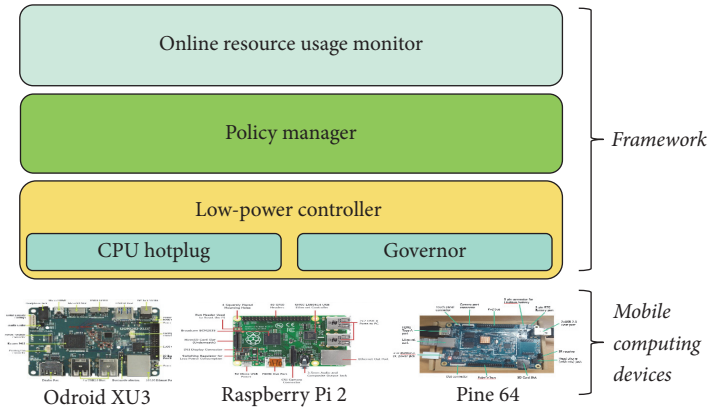


Figure 1: Structure of the proposed low-power framework.

Zhu et al. observe that the wakeup delay of the sleeping cores takes hundreds of microseconds, which is at least 100 times slower than the frequency change delay [20]. To hide this latency, they devise an anticipatory CPU wakeup for maintaining high performance. Song et al. propose a framework that applies low-power techniques based on user-perceived response time analysis [21].

Wamhoff et al. design a library that assigns heterogeneous and even boosted frequencies for accelerating performance [22]. Chiesi et al. present a power-aware scheduling algorithm on heterogeneous architectures to reduce the peak power [23].

DPM and DVFS are also studied in the real-time research area [10, 24, 25]. Bambagini et al. present a survey paper that discusses various energy-aware scheduling algorithms for real-time mobile systems [24]. Li and Broekaert design a DVFS based low-power scheduler that exploits slack times with an intratask intrusive approach [25]. Chen et al. devise a technique that models the idle intervals of individual cores to optimize the DVFS and DPM for real-time tasks [10].

2.4. System/Application-Level Approach. Roy et al. propose a new operating system, called *cinder*, for mobile phones and energy-constrained computing devices [26]. It supports two new abstractions, *reserves* and *taps*, which store and distribute energy for applications. Snowdon et al. design a framework; they refer to it as *Koala*, which provides a model, a generalized energy-delay policy, and a single parameter for tuning the system to an overall energy-management objective [27].

Shen et al. present a new operating system facility, called *power containers*, that controls the power and energy usage of individual fine-grained requests in multicore systems [28]. Zhu and Shen observe the energy disproportionality in multicore devices, where the first running CPU incurs much higher power cost than each additional core does [6]. Kwon et al. propose a framework that predicts the computational resource consumption on mobile devices using program analysis and machine learning [29].

Thiagarajan et al. design an infrastructure for measuring energy used by a mobile browser to render web pages such as email, e-commerce, and social networking sites [30]. Using the infrastructure, they observe that downloading and parsing CCS (Cascade Style Sheets) and JavaScript consume a large portion of energy and recommend how to design web pages for enhancing energy efficiency. Bui et al. propose techniques, namely, adaptive content painting and application-assisted scheduling, to improve the energy efficiency of web page loading [31].

3. Adaptive and Integrated Low-Power Framework

In this section, we first explain the overall structure of our framework and principle used to design the framework. Then, we discuss the details of each component in sequence.

3.1. Overall Structure. Figure 1 displays the overall structure of the adaptive and integrated low-power framework proposed in this paper. It consists of three components, called online resource usage monitor, policy manager, and low-power controller.

When we design our framework, we use the design principle that separates policy and mechanism to support diverse policies flexibly. The policy manager takes charge of the policy decision, while the low-power controller provides mechanisms for DPM and DVFS. The online resource monitor is used for supporting adaptability based on resource usage patterns including the CPU utilization and power consumption.

Currently, the framework works on three multicore-based mobile computing devices, namely, Odroid XU3 [32], Raspberry Pi2 [33], and Pine 64 [34]. Note that the framework can be installed any Linux-based devices since it utilizes the standard Linux interfaces only.

3.2. Low-Power Controller. The low-power controller component provides mechanisms for DPM and DVFS. Specifically, it makes use of the CPU hotplug mechanism [16] for DPM and the governor mechanism [17] for DVFS as shown in Figure 1.

The CPU hotplug is a mechanism that turns on and off an individual core dynamically. It was originally designed to

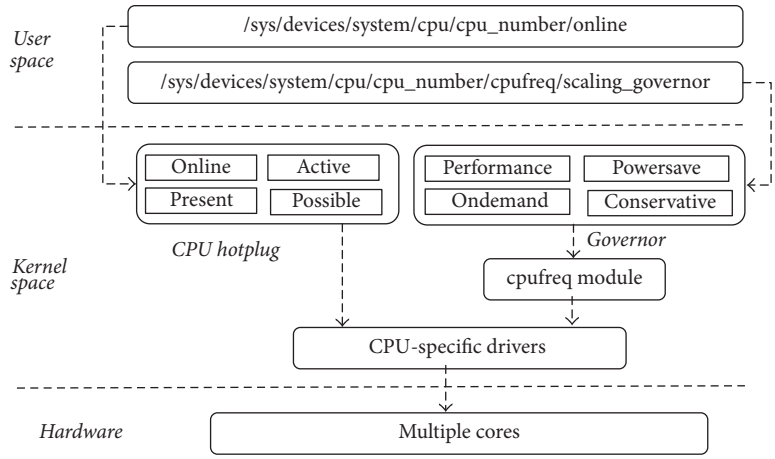


Figure 2: Internals of the low-power controller.

allow a failing core to be removed from a running system, but now it is popularly used for energy management.

Figure 2 shows the internal behavior of the low-power controller. The CPU hotplug mechanism supports a file, /sys/devices/system/cpu/cpu_number/online file, into user space using the *sysfs* file system that is a pseudo file system exporting various kernel information in Linux. A user can turn on or off a core by writing 1 or 0 to the file.

While turning on or off a core, the core goes through various states such as possible, present, active, and online [35]. At each state, the mechanism invokes various functions provided by CPU-specific drivers such as cpu_up(), cpu_online(), cpu_down(), and cpu_down_prepare(). Finally, the on/off request is applied into the designated core at hardware level.

The most time consuming job in the CPU hotplug mechanism is the context management. To turn off a core, it needs to save the context of a process that runs on the core and migrates the context to another core to continue its execution. Also, to turn on a core, it has to prepare a new scheduling queue for the core to be active. Hence, the latency for a core on/off is much higher than that for changing frequency of a core. We need to consider this difference for our integrated low-power management.

For DVFS, the low-power controller makes use of the governor mechanism. It allows changing the frequency of a core dynamically. It supports a file, /sys/devices/system/cpu/cpu_number/cpufreq/scaling_governor, into user space as shown in Figure 2.

Linux employs several default governors, namely, performance, powersave, ondemand, conservative, and userspace [17]. The distinctions of these governors are illustrated in Figure 3. In the performance governor, a core always runs at the maximum frequency. On the contrary, in the powersave governor, a core runs at the minimum frequency.

In the ondemand governor, a core runs initially at the minimum frequency. When a CPU load increases, the frequency becomes the maximum frequency immediately to minimize the effect of DVFS on the application execution time. When the load decreases, the frequency goes down gradually into the minimum frequency. In the conservative governor, a core runs initially at the minimum frequency like the ondemand governor. But, when the load becomes intensive, the frequency increases gradually. Finally, the userspace governor allows any frequency to be set by a user.

Commercial smartphones extend these governors, devising their own specific governors. For instance, Android uses a governor, called interactive governor, which behaves similar to the ondemand governor but responds more quickly [15].

Each governor is based on a subsystem, called *cpufreq*, which provides interfaces to explicitly set frequency on cores. This subsystem eventually makes use of the CPU-specific drivers that actually implement CPU on/off and frequency change functionality for various vendors including ARM, Intel, and AMD.

The low-power controller integrates the CPU hotplug and governor mechanisms and provides virtual interfaces such as increase_computing() and decrease_computing(). These interfaces invoke the mechanisms appropriately based on the decision dictated by the policy manager, which will be further discussed in Section 3.4.

3.3. Online Resource Usage Monitor. The online resource usage monitor gives information about how much resources are used by the current workload so that our framework can apply the low-power mechanisms adaptively. It makes use of the Linux *proc* file system, measuring various resource usage statistics such as CPU utilization, memory footprint, power consumption, and process activities. In addition, it reports the measured statistics via a web page using Node.js.

Figure 4 presents the CPU utilization statistics measured by the monitor on the Odroid XU3 device. This device has the big.LITTLE architecture, consisting of four little cores (ARM Cortex-A7) and four big cores (ARM Cortex-A15). The figure shows that the first little core (CPU0) runs at the frequency of 1.4 GHz whose utilization for user, system, and idle state is 6.79%, 3.61%, and 88.72%, respectively.

To measure the power consumption, the monitor makes use of the power measurement functionality

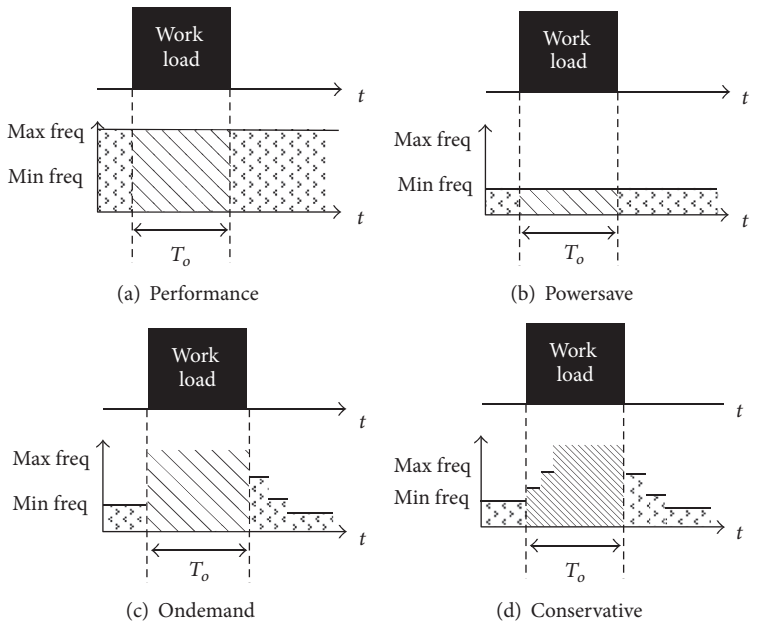
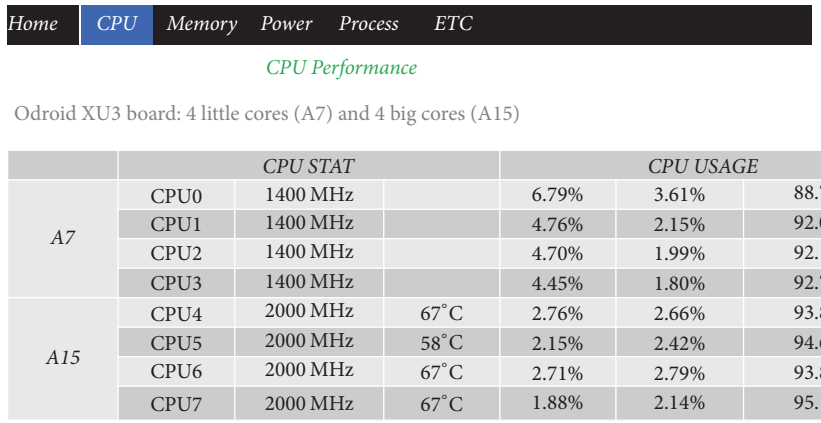


Figure 3: Frequency changes in various governors.



	CPU STAT			CPU USAGE		
A7	CPU0	1400 MHz		6.79%	3.61%	88.72%
	CPU1	1400 MHz		4.76%	2.15%	92.07%
	CPU2	1400 MHz		4.70%	1.99%	92.17%
	CPU3	1400 MHz		4.45%	1.80%	92.74%
A15	CPU4	2000 MHz	67°C	2.76%	2.66%	93.81%
	CPU5	2000 MHz	58°C	2.15%	2.42%	94.61%
	CPU6	2000 MHz	67°C	2.71%	2.79%	93.88%
	CPU7	2000 MHz	67°C	1.88%	2.14%	95.12%

Figure 4: CPU utilization reported by the online resource usage monitor on the Odroid XU3 device.

already equipped in a mobile device. The Odroid XU3 device supports such functionality, providing the power consumption of each unit including big core, little core, GPU, and RAM. For devices that do not support such functionality, Raspberry Pi 2 and Pine 64 in this paper, we utilize an external power meter that can quantify the overall power consumption by assessing the voltage driven into the power unit.

3.4. Policy Manager. Based on the measured information provided by the online resource usage monitor, the policy manager makes a policy that can lead to better energy efficiency. Then, it enforces the policy using interfaces supported by the low-power controller.

The policy manager introduces four control parameters, namely, *max_utilization*, *min_utilization*, *max_frequency*, and *min_frequency*. The former two parameters are used to trigger the policy manager to enforce its policy while the latter two parameters are used for DVFS. For instance, when the current utilization of a core is higher than the max_utilization, the policy manager is triggered to increase computing resource.

Since our framework integrates two techniques, DPM and DVFS, we need to have a rule about which technique is applied first. The policy manager supports two options. The first option is providing an interface so that a user can specify his/her preference. The second option is giving a default rule by considering the overhead of the two techniques and features of mobile devices.

To devise a guideline for the default rule, we analyze the overhead of the techniques and the power reduced by them using the online resource usage monitor. We make the

```
decrease_computing()
for each big core do
  if (core.freq > parameter.big.min_freq) then
    decrease core.freq to the next lower level
    return
for each big core do
  if (core.state == on) then
    turn off this core
    return
for each little core do
  if (core.freq > parameter.little.min_freq) then
    decrease core.freq to the next lower level
    return
for each little core do
  if (core.state == on) then
    turn off this core
    return
```

Algorithm 1: Pseudocode for decreasing computing resources in the policy manager.

```
increase_computing()
for each little core do
  if (core.state == off) then
    turn on this core
    return
for each little core do
  if (core.freq < parameter.little.max_freq) then
    increase core.freq to the next higher level
    return
for each big core do
  if (core.state == off) then
    turn on this core
    return
for each big core do
  if (core.freq < parameter.big.max_freq) then
    increase core.freq to the next higher level
    return
```

Algorithm 2: Pseudocode for increasing computing resources in the policy manager.

following three observations. First, the latency to turn off a core is much longer than that to change the frequency of a core. Second, the power saved by turning off a core is smaller than that by decreasing the frequency of a core. This result is also observed by Carroll and Heiser's study where they recommend that one should "offline cores conservatively and reduce frequency aggressively" [19]. This is partly because cores share various resources in the same processor [6]. Third, the power used by big cores is much higher than that by little cores.

Our observations lead us to design an algorithm, shown in Algorithm 1, which is invoked when we decrease computing resource. The algorithm prefers big cores to little ones and prefers DVFS to DPM by default. Specifically, it first tries to reduce the frequency among big cores. If it is not possible, it tries to turn off a big core. Again, if not feasible, it tries to reduce the frequency and to turn off a core among little cores. When one of the four "if" conditions satisfies, this function returns without going further.

Algorithm 2 presents an algorithm for increasing computing resources. The sequence of this algorithm is reverse to that of the one in Algorithm 1. It gives a higher priority for little cores. Then, it tries to turn on a core before increasing the frequency. The next higher level and the next lower level in the pseudocodes are determined by the governors, discussed in Figure 3.

The policy manager provides a configuration file so that a user can configure his/her preferred parameters. The parameters include min/max utilization, min/max frequency, DPM/DVFS preference, monitoring period, and monitoring number. The default values for the preference, monitoring period, and monitoring number are DPM preference, 200 milliseconds, and 5, respectively. The monitoring number is the number of consecutive measurements for calculating the moving average of the CPU utilization. When it is small, the policy tries to apply the low-power techniques aggressively. When a user prefers DVFS, the first/third part of the pseudocodes is changed with the second/fourth part.

Table 1: Multicore-based mobile devices.

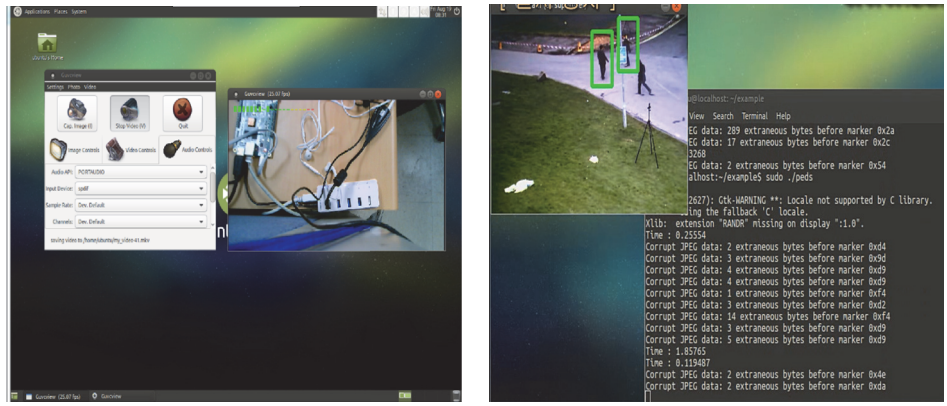
Device	Multicore description
Odroid XU3 [32]	Octa cores (Cortex-A15 4 cores, Cortex-A7 4 cores)
Raspberry Pi 2 [33]	Quad cores (Cortex-A7 4 cores)
Pine 64 [34]	Quad cores (Cortex-A53 4 cores)

4. Evaluation

In this section, we first describe our experimental devices and workloads considered in this paper. Then, we discuss evaluation results from the power measurements to the power and energy saving achieved by the proposed framework.

4.1. Experimental Environments. We have implemented the framework on the Linux kernel version 3.10. The online resource usage monitor uses the proc file system for monitoring and reports usage statistics via a web page using Node.js. The low-power manager provides the integrated low-power interfaces based on the CPU hotplug and governor mechanism. The policy manager is implemented as a daemon process that analyzes resource usage statistics at every 200 milliseconds and applies the integrated low-power interfaces when the current CPU utilization is above/below the threshold of the max/min utilization, whose default values are 80% and 20% in this experiment.

Table 1 summarizes three mobile computing devices used in this study. The Odroid XU3 device has the big.LITTLE



(a) Video recording (b) Pedestrian detection

Figure 5: Workloads for experiments.

architecture, consisting of four little cores (Cortex-A7) and four big cores (Cortex-A15). Each core supports the same ISA and equips L1, L2 cache, where the size of L2 cache for little and big core is 512 KB and 2 MB, respectively. Both the Raspberry Pi 2 and Pine 64 devices have homogeneous four cores, Cortex-A7 and Cortex-A53, respectively. Note that the multicore architecture used in Odroid XU3, also called Exynos 5422, is actually employed for commercial mobile devices including Samsung Galaxy S5 and Chromebook 2.

We use three workloads for experiments. The first one is a video recording, as shown in Figure 5(a). It is an I/O intensive workload, recording using a camera and displaying through LCD. The second one is the Sunspider test suite [36]. It is a CPU intensive workload that tests JavaScript performance including function calls, math, and recursion without rendering. The third one is a pedestrian detection using the Haar classifier [37].

4.2. Evaluation Results

4.2.1. Video Recording Workload Results. This section consists of two parts. In the first part, we explain evaluation results observed using the Odroid XU3 device that has heterogeneous 8 cores. The second part is the results of the Raspberry Pi2 and Pine 64 device that have homogeneous 4 cores. Note that, in Odroid XU3, we can measure the power consumption of each component individually using the measurement functionality already equipped in the device while, in Raspberry Pi2 and Pine 64, we only measure the overall power consumed by the device using the external power meter, as discussed in Section 3.3.

Figure 6 presents the power measurement results of the Odroid XU3 device when it is in an idle state. The results are reported by the online resource usage monitor discussed in Section 3.3. This measurement is conducted under the baseline configuration where all hardware components are powered on. The results reveal that big cores are the heaviest power consumer, using 0.929 watt, while little cores, GPU, and DRAM consume 0.155, 0.055, and 0.096 watt, respectively. Note that big cores consume quite large power even in an idle state, demonstrating the importance of the DPM and DVFS techniques.

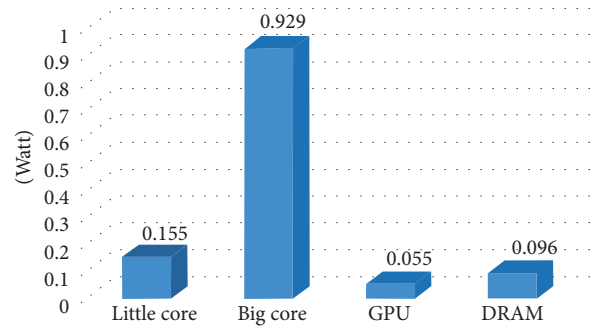


Figure 6: Power measurement results under the idle state on the Odroid XU3 device.

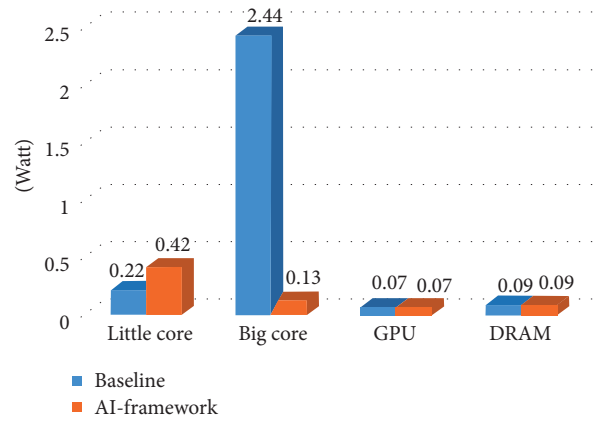


Figure 7: Power consumption comparison under the video recording workload on the Odroid XU 3 device.

To evaluate the power consumption reduced by our proposed framework, we execute the video recording workload under the two configurations, as shown in Figure 7. The first configuration is the baseline where any low-power

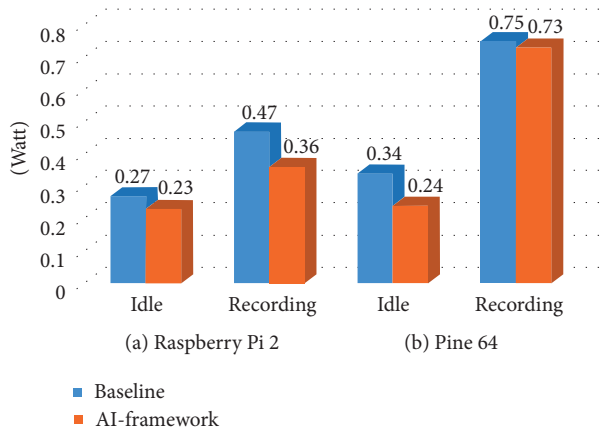


FIGURE 8: Power consumption comparison on the Raspberry Pi 2 and Pine 64 device.

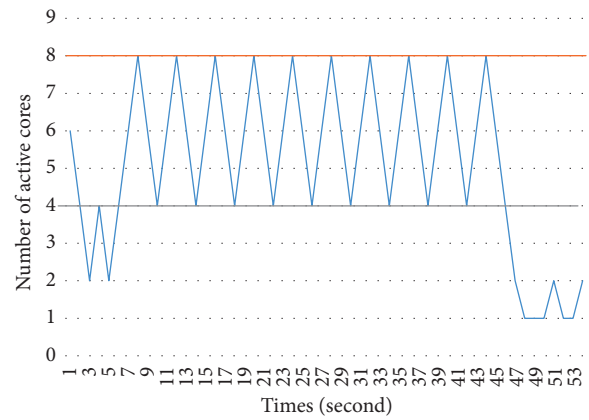


FIGURE 9: The number of active cores when we execute the Sunspider workload on the Odroid XU3 device.

technique is not applied. The second configuration is under our framework, labelled as AI-framework (Adaptive and Integrated framework) in the figure.

Since the video recording is not a CPU intensive workload, it uses only one core during most of its execution period. Hence, our framework turns on one little core, while turning off other cores. On the contrary, in the baseline, all cores are powered on and the workload mostly runs on one of the big cores. As a result, the power consumed by big cores becomes 2.44 watt in the baseline, while that consumed in the AI-framework is 0.13 watt due to offlining. The power consumed by little cores is 0.42 watt in the AI-framework, higher than that in the baseline. The overall power consumed by all cores in the baseline is 2.66 watt while that consumed in the AI-framework is 0.55 watt (79% reduction).

Figure 8 presents the power measurement results using Raspberry Pi 2 and Pine 64 under the idle and the video recording case. Note that these devices provide the frequency change functionality only, not supporting the dynamic power off functionality for an individual core. Hence, in this experiment, the AI-framework exploits the DVFS technique only.

The results show that our proposed AI-framework can reduce the power consumption by decreasing CPU frequency appropriately. For Raspberry Pi2, it reduces the power consumption from 0.27 to 0.23 watt under the idle state and from 0.47 to 0.36 watt when we run the video recording workload. For Pine 64, it reduces from 0.34 to 0.24 watt and from 0.75 to 0.73 watt, respectively. The reduction is relatively low for the video recording workload on the Pine 64 device. We conjecture that the camera module equipped in the Pine 64 device consumes a large portion of power consumption, leading to this small difference. We leave the component-level fine-grained power analysis as the future work.

4.2.2. Sunspider Workload Results. Figure 9 presents the number of active cores that are powered on during the execution period of the Sunspider workload in the AI-framework on the Odroid XU3 device. It shows that when the workload requires a large computing resource, the number of active cores increases up to the maximum cores. When

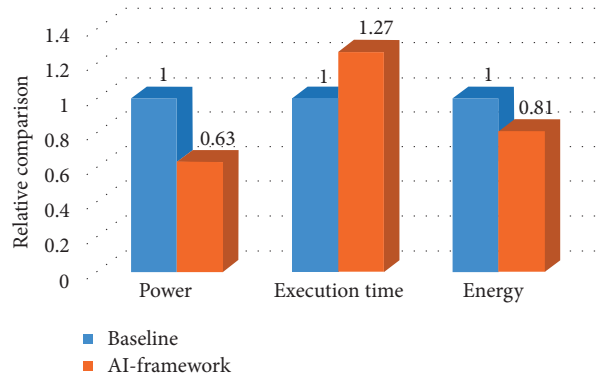


FIGURE 10: Power consumption, execution time, and energy saving comparison using the Sunspider workload on the Odroid XU 3 device.

the workload does not need that much computing resource, the active cores decrease down to the one core. It reveals that our framework indeed supports adaptability according to the workload characteristics.

Figure 10 presents the power consumption, execution time, and consumed energy when we run the Sunspider workload under the baseline and AI-framework. Note that the y-axis is the relative value. The power consumed in the baseline is 2.6 watt while that in the AI-framework is 1.6 watt (37% reduction).

However, the execution time of the workload in the baseline is 39 seconds while that in the AI-framework is 49 seconds (27% degradation). It shows the tradeoff of the low-power techniques, reducing power consumption at the expense of performance drop. As a net result, the AI-framework can achieve the 19% energy saving, reducing from 1.97 to 1.6 joule. For a CPU intensive workload, we can mitigate the performance drop by turning off cores conservatively, which will be further discussed in the next section.

We also run the Sunspider workload on the Pine 64 and Raspberry Pi 2 device. The results show that even though the

Table 2: Power consumption and detection latency on the Pine 64 and Raspberry Pi 2 device.

Device	Configuration	Power (watt)	Latency (ms)
Pine 64	AI-framework	2.47	122.7
	Baseline 1: max freq	3.18	121.6
	Baseline 2: min freq	2.07	275.0
Raspberry Pi2	AI-framework	1.34	239.1
	Baseline 1: max freq	1.41	230.5
	Baseline 2: min freq	1.28	338.1

Table 3: Power consumption and detection latency on the Odroid XU3 device (max utilization: 80%, min utilization: various).

Device	Configuration	Power (watt)	Latency (ms)
Odroid XU3	AI-framework: min util = 0%	5.90	251.7
	AI-framework: min util = 10%	5.74	264.8
	AI-framework: min util = 30%	4.38	267.4
	AI-framework: min util = 60%	3.13	425.2

AI-framework provides better energy efficiency as discussed in Figure 10, the improvement is small, ranging from 1% to 6%. Our analysis reveals that since the Sunspider workload is CPU intensive, requiring more than 4 cores on average, the AI-framework does not have enough chance to apply DVFS. Note that these two devices have 4 cores, as explained in Table 1.

4.2.3. Pedestrian Detection Workload Results. We measure the power consumption and average detection latency when we execute the pedestrian detection workload on the Pine 64 and Raspberry Pi2 device, presented in Table 2. Since we can utilize DVFS only in these devices, we conduct experiments under three configurations. In the baseline 1, we configure all cores to run at the maximum frequency (1152 MHz for Pine 64 and 900 MHz for Raspberry Pi2). In baseline 2, all cores are configured to run at the minimum frequency (480 MHz for Pine 64 and 200 MHz for Raspberry Pi2). On the contrary, in the AI-framework, the frequency of a core is changed adaptively based on the current CPU utilization and the min/max utilization threshold (20% and 80% in this experiment).

Experimental results show that the AI-framework balances well between the power consumption and performance. Baseline 1 provides the best performance at the cost of high power consumption. On the contrary, baseline 2 reduces power the most but gives a noticeable impact on performance. However, our framework can reduce the power consumption (22% reduction for Pine 64 and 5% reduction for Raspberry Pi2) while hardly affecting the performance of the workload.

Table 3 shows the results when we execute the pedestrian detection workload on the Odroid XU3 device. In this device, the AI-framework can utilize not only DVFS but also DPM. Therefore, we conduct experiments with four different *min_utilization* threshold values that trigger the policy manager in our framework as discussed in Section 3.4. When the threshold becomes smaller, the AI-framework tries to apply low-power techniques conservatively, while applying techniques aggressively as the threshold becomes larger.

When the min utilization threshold is set as 0%, the AI-framework tries to decrease computing resources when the current utilization is less than 0%. It means that the AI-framework does not apply DPM and DVFS, turning on all cores with maximum frequency, which provides the best performance and the worst power consumption in this device (baseline configuration). When the threshold is 10%, the AI-framework tries to decrease computing resources conservatively, obtaining relatively small power reduction (from 5.9 to 5.74 watt in this case). On the contrary, when the threshold is 60%, it tries aggressively, yielding better power reduction at the cost of latency. These results reveal the tradeoff between power reduction and performance. By setting the threshold appropriately (30% in this case), the AI-framework can reduce the power consumption without considerable performance degradation.

5. Conclusion

In this paper, we design a new low-power framework for multicore mobile devices. It integrates both DPM and DVFS techniques and applies them adaptively according to the workload characteristics and device features. Real implementation based experiments show that the proposed framework balances well between the power consumption and performance, resulting in the energy saving.

We will extend our work into the two directions. First, we investigate the performance drop, especially for a CPU intensive workload observed in Figure 10, using hardware-level performance monitoring unit supported by processors. We conjecture that workload-aware fine-grained power management can alleviate the drop while maintaining the power reduction benefit. The second direction is developing a what-if engine that can predict how an alteration of frequency or number of active cores influences energy efficiency in advance using our framework.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The present research was conducted by the research fund of Dankook University (BK21 Plus) in 2014 and by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2016-R0992-16-1012) supervised by the IITP and by the Center for Integrated Smart Sensors funded by

the Ministry of Science, ICT & Future Planning as Global Frontier Project (CISS-2-3).

References

[1] N. D. Lanez, S. Bhattacharyaz, P. Georgievy, C. Forlivesiz, and F. Kawsar, "An early resource characterization of deep learning on wearables, smartphones and internet-of-things devices," in *Proceedings of the ACM International Workshop on Internet of Things towards Applications (IoT-App '15)*, November 2015.

[2] J. S. Kim, D. H. Yeom, and Y. H. Joo, "Fast and robust algorithm of tracking multiple moving objects for intelligent video surveillance systems," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 3, pp. 1165–1170, 2011.

[3] R. Kumar, D. M. Tullsen, P. Ranganathan, N. P. Jouppi, and K. I. Farkas, "Single-ISA heterogeneous multi-core architectures for multithreaded workload performance," *ACM SIGARCH Computer Architecture News*, vol. 32, no. 2, 2004.

[4] C. Baun, "Mobile clusters of single board computers: an option for providing resources to student projects and researchers," *SpringerPlus*, vol. 5, no. 1, article 360, 2016.

[5] T. Guan, Y. Wang, L. Duan, and R. Ji, "On-device mobile landmark recognition using binarized descriptor with multi-feature fusion," *ACM Transactions on Intelligent Systems and Technology*, vol. 7, no. 1, article 12, 2015.

[6] M. Zhu and K. Shen, "Energy discounted computing on multicore smartphones," in *Proceedings of the USENIX Annual Technical Conference (ATC '16)*, Denver, Colo, USA, June 2016.

[7] A. Carroll and G. Heiser, "Unifying DVFS and offlining in mobile multicores," in *Proceedings of the 20th IEEE Real Time and Embedded Technology and Applications Symposium (RTAS '14)*, pp. 287–296, April 2014.

[8] Y. Tawara, A. Idehara, and H. Yamamoto, "DVFS and power-off controls on a multicore operating system," in *Proceedings of the 10th International Forum on Embedded MPSoC and Multicore (MPSoC '10)*, Gifu, Japan, June 2010.

[9] J. M. Kim, Y. G. Kim, and S. W. Chung, "Stabilizing CPU frequency and voltage for temperature-aware DVFS in mobile devices," *IEEE Transactions on Computers*, vol. 64, no. 1, pp. 286–292, 2015.

[10] G. Chen, K. Huang, and A. Knoll, "Energy optimization for real-time multiprocessor system-on-chip with optimal DVFS and DPM combination," *ACM Transactions on Embedded Computing Systems*, vol. 13, no. 3s, article 111, 2014.

[11] L. Benini, A. Bogliolo, and G. De Micheli, "A survey of design techniques for system-level dynamic power management," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 8, no. 3, pp. 299–316, 2000.

[12] M. E. Salehi, M. Samadi, M. Najibi, A. Afzali-Kusha, M. Pedram, and S. M. Fakhraie, "Dynamic voltage and frequency scheduling for embedded processors considering power/performance tradeoffs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 10, pp. 1931–1935, 2011.

[13] C. Gao, A. Gutierrez, M. Rajan, R. G. Dreslinski, T. Mudge, and C.-J. Wu, "A study of mobile device utilization," in *Proceedings of the 15th IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS '15)*, pp. 225–234, March 2015.

[14] Y. Zhang, X. Wang, X. Liu, Y. Liu, L. Zhuang, and F. Zhao, "Towards better CPU power management on multicore smartphones," in *Proceedings of the ACM Workshop on Power-Aware Computing and Systems (HotPower '13)*, Farmington, Pa, USA, November 2013.

[15] W. Seo, D. Im, J. Choi, and J. Huh, "Big or little: a study of mobile interactive applications on an asymmetric multi-core platform," in *Proceedings of the IEEE International Symposium on Workload Characterization (IISWC '15)*, pp. 1–11, IEEE, Atlanta, Ga, USA, October 2015.

[16] Z. Mwaikambo, A. Raj, R. Russell, J. Schopp, and S. Vaddagiri, "Linux kernel CPU hotplug support," in *Proceedings of the OLS*, July 2004.

[17] V. Pallipadi and A. Starikovskiy, "The ondemand governor," in *Proceedings of the Ottawa Linux Symposium (OLS '06)*, Ottawa, Canada, July 2006.

[18] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proceedings of the USENIX Conference on USENIX Annual Technical Conference (USENIXATC '10)*, ACM, Boston, Mass, USA, 2010.

[19] A. Carroll and G. Heiser, "Mobile multicores: use them or waste them," in *Proceedings of the Workshop on Power-Aware Computing and Systems (HotPower '13)*, November 2013.

[20] Q. Zhu, M. Zhu, B. Wu, X. Shen, K. Shen, and Z. Wang, "Software engagement with sleeping CPUs," in *Proceedings of the 15th Workshop on Hot Topics in Operating Systems (HotOS '15)*, Kartause Ittingen, Switzerland, March 2015.

[21] W. Song, N. Sung, B.-G. Chun, and J. Kim, "Reducing energy consumption of smartphones using user-perceived response time analysis," in *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications (HotMobile '14)*, ACM, February 2014.

[22] J. Wamhoff, S. Diestelhorst, C. Fetzer, P. Marlier, P. Felber, and D. Dice, "The TURBO diaries: application-controlled frequency scaling explained," in *Proceedings of the USENIX Annual Technical Conference (USENIX ATC '14)*, June 2014.

[23] M. Chiesi, L. Vanzolini, C. Mucci, E. Franchi Scarselli, and R. Guerrieri, "Power-aware job scheduling on heterogeneous multicore architectures," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 3, pp. 868–877, 2015.

[24] M. Bambagini, M. Marinoni, H. Aydin, and G. Buttazzo, "Energy-aware scheduling for real-time systems: a survey," *ACM Transactions on Embedded Computing Systems*, vol. 15, no. 1, article 7, 2016.

[25] S. Li and F. Broekaert, "Low-power scheduling with DVFS for common RTOS on multicore platforms," in *Proceedings of the 3rd Embedded Operating Systems Workshop (EWiLi '13)*, Toulouse, France, August 2013.

[26] A. Roy, S. M. Rumble, R. Stutsman, P. Levis, D. Mazières, and N. Zeldovich, "Energy management in mobile devices with the Cinder operating system," in *Proceedings of the 6th ACM EuroSys Conference on Computer Systems (EuroSys '11)*, pp. 139–152, April 2011.

[27] D. C. Snowdon, E. L. Sueur, S. M. Petters, and G. Heiser, "Koala: a platform for OS-level power management," in *Proceedings of the EuroSys*, March-April 2009.

[28] K. Shen, A. Shriraman, S. Dwarkadas, X. Zhang, and Z. Chen, "Power containers: an OS facility for fine-grained power and energy management on multicore servers," in *Proceedings of the 18th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '13)*, pp. 65–76, ACM, March 2013.

[29] Y. Kwon, S. Lee, H. Yi et al., "Mantis: efficient predictions of execution time, energy usage, memory usage and network usage on smart mobile devices," *IEEE Transactions on Mobile Computing*, vol. 14, no. 10, pp. 2059–2072, 2015.

[30] N. Thiagarajan, G. Aggarwal, A. Nicoara, D. Boneh, and J. P. Singh, "Who killed my battery: analyzing mobile browser energy consumption," in *Proceedings of the 21st International Conference on World Wide Web (WWW '12)*, pp. 41–50, ACM, Lyon, France, April 2012.

[31] D. H. Bui, Y. Liu, H. Kim, I. Shin, and F. Zhao, "Rethinking energy-performance trade-off in mobile web page loading," in *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (MobiCom '15)*, pp. 14–26, ACM, Paris, France, September 2015.

[32] http://www.hardkernel.com/main/products/prdt_info.php?g_code=g140448267127.

[33] https://www.raspberrypi.org/products/raspberry-pi-2-model-b/.

[34] https://www.pine64.org/.

[35] CPU hotplug Support in Linux Kernel, https://lwn.net/Articles/537570/.

[36] J. Resig, "JavaScript Performance Rundown," http://ejohn.org/blog/javascript-performance-rundown/.

[37] P. I. Wilson and J. Fernandez, "Facial feature detection using HAAR classifiers," *Journal of Computing Sciences in Colleges*, vol. 21, no. 4, pp. 127–133, 2006.

Hindawi
Mobile Information Systems
Volume 2017, Article ID 3273917, 17 pages
https://doi.org/10.1155/2017/3273917

Research Article

SACA: Self-Aware Communication Architecture for IoT Using Mobile Fog Servers

Vishal Sharma,[1] Jae Deok Lim,[2] Jeong Nyeo Kim,[2] and Ilsun You[1]

[1]*The Department of Information Security Engineering, Soonchunhyang University, Asan-si 31538, Republic of Korea*
[2]*Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea*

Correspondence should be addressed to Ilsun You; ilsunu@gmail.com

Received 19 January 2017; Revised 19 February 2017; Accepted 26 February 2017; Published 11 April 2017

Academic Editor: Eric Rondeau

Internet of things (IoT) aims at bringing together large business enterprise solutions and architectures for handling the huge amount of data generated by millions of devices. For this aim, IoT is necessary to connect various devices and provide a common platform for storage and retrieval of information without fail. However, the success of IoT depends on the novelty of network and its capability in sustaining the increasing demand by users. In this paper, a self-aware communication architecture (SACA) is proposed for sustainable networking over IoT devices. The proposed approach employs the concept of mobile fog servers which make relay using the train and unmanned aerial vehicle (UAV) networks. The problem is presented based on Wald's maximum model, which is resolved by the application of a distributed node management (DNM) system and state dependency formulations. The proposed approach is capable of providing prolonged connectivity by increasing the network reliability and sustainability even in the case of failures. The effectiveness of the proposed approach is demonstrated through numerical and network simulations in terms of significant gains attained with lesser delay and fewer packet losses. The proposed approach is also evaluated against Sybil, wormhole, and DDoS attacks for analyzing its sustainability and probability of connectivity in unfavorable conditions.

1. Introduction

Sustainable communication is one of the key demands of network devices. With a large number of network devices making continuous requests, it becomes important to provide architectural support for continuous connectivity. With a tremendous growth in the number of devices already observed and expected in the near future, sustainable and context-aware communication is in high demand [1].

Internet of things (IoT) has bridged the gap between the network technology and the devices operating over it. Most of the devices are able to utilize the network as a service for sharing data between them. Availability and easy access are the critical issues to be handled with a high number of devices operating on a common platform for service capturing [2, 3]. Infrastructure and network play a key role in providing services to all the devices. A novel architecture can enhance the session quality and can provide prolonged connectivity even in the nonsupporting conditions such as failures, network breakdowns, or security breaches.

With a demand of efficient data management, continuous connectivity, security, and context-aware service provisioning, it becomes important to provide an architecture which can maintain connections between nodes [4, 5]. Most of the authors have defined sustainability according to the domain and area of application such as IoT for water management [6]. Use of artificial intelligence and machine learning approaches can be two of the key solutions for sustainable IoT. Efficient networks can provide enhanced management and control over the devices, whereas a slight irregularity in the network can make it vulnerable to many issues such as privacy, trust, and session hijacking [7–9]. A novel network can provide a common solution for sustainable IoT and can form the backbone of most of the approaches. There are several approaches that aim at the formation of sustainable IoT by considering the service-level solutions, which are specific to a particular domain. However, the success of such approaches depends on the novelty in network layouts. This is an inefficient way of enhancing connectivity until the underlying network is not robust enough to support the service solutions. Thus,

in order to overcome the issue of network breakdowns for sustainable IoT, a novel architecture is proposed in this paper, which forms an intelligent solution for node management. The proposed approach uses a key concept of fog computing but in a novel way by placing fog servers on a train network.

Fog computing/fogging is an innovative nomenclature given to the near user cloud to reduce the latency involved in the flow of data [10]. With major of its properties derived from cloud computing, fogging also utilizes the key concept of mobile clouds [11]. However, it should not be confused with the mobile cloud operations, since mobile cloud aims at utilizing the mobile devices as a platform for implementing cloud applications which are handled as batches, whereas, in this paper, mobile fog computing refers to placing fog servers on fast-moving platforms which can handle the demand for continuous connectivity irrespective of the number of users.

A multitier architecture is proposed in this paper, which provides a self-aware communication setup for handling services across the network. The proposed approach uses unmanned aerial vehicles (UAVs) as intermediate flying routers between the fixed on-ground nodes to provide immediate and efficient connectivity. UAVs can fly in controlled as well as autonomous formations [12, 13]. UAVs have already proven their utility in the upcoming networks and can provide coverage over the large areas [14–17]. Although server computations can be performed on the aerial vehicles, this requires consideration of payload which is a constraint in the utilization of UAVs for operations that require heavy equipment.

Train networks form the key part of the proposed solution which is the actual near user site for placing the fog servers and is an intermediate between the user network and the core network. A distributed node management (DNM) module is used by station terminals for managing all the network nodes considering a state diagram which is defined over the order of connectivity.

The proposed self-aware communication architecture (SACA) is a multimodular hybrid network approach which utilizes the train network and UAV network to exploit the service-guaranteeing features of upcoming 5G networks. Further, sensor feeding, sensor signatures, and content-based server allocation policies are used for managing the load by forming optimization problems using Wald's maximum model [18]. The key contributions of the proposed solution are listed as follows:

(i) A novel hybrid architecture comprising train and UAV networks using the concept of fogging

(ii) Sensor signatures and content-based server allocation for load balancing

(iii) Highly reliable and sustainable network formation even during node/link failures as well as during network attacks

(iv) Intelligent decision-making in the case of network threats and attacks using network state dependency and DNM

The rest of the paper is organized as follows: Section 2 presents the related work. Section 3 presents the motivation and problem statement. Section 4 gives the details of proposed work along with the theoretical analyses. Section 5 evaluates the performance of the proposed approach. Section 6 presents discussions, open issues, and comparison with the existing state-of-the-art approaches. Finally, Section 7 concludes the paper.

2. Related Work

Internet of things aims at providing connectivity to all and connectivity on the go. With a large number of devices generating a huge amount of data and service requests, it becomes important to provide a sustainable strategy that can withstand such tremendous demand for connectivity. Over the years, a lot of attempts have been made for designing sustainable architectures and models to support a large number of IoT devices.

2.1. Sustainable IoT. IoT has made life much easier for humans but complex for the devices and technology handling it. Handling a large number of requests, data dropouts, and security issues and connection stability are the key metrics for defining the sustainable IoT [2, 19]. Many architectures and models exist which have utilized one or other features to provide prolonged connectivity between the IoT devices and network infrastructure. Some of them have focused on device-to-device approach [20], while others emphasized on device-to-infrastructure-to-device methodology [21, 22].

Riedel et al. [23] used web-service gateways along with the code generation to enhance the sustainability of IoT networks. However, depending heavily on the client side gateway can add up to the issues of congestion in a network comprising a large number of simultaneously operating devices. Designing of efficient systems on a chip for providing energy efficient connectivity can also provide sustainable IoT [1]. El Kaed et al. [4] developed a semantic query system for industrial IoT. The authors utilized the concept of semantic tagging of products for making a sustainable IoT for industrial applications. However, the primary focus of their approach is in the selection of gateways which ignores other key factors such as reliability and fault-tolerance.

2.2. Sustainable Fogging. Cloud and fog computing based IoT can provide a vast range of applications using the service selection strategies [24, 25]. The operability of an efficient fog computing environment depends on the efficient policy formation [26]. With a focus on the application-oriented near user cloud formation, policy driver architectures can provide sustainable connectivity. Embedding existing wireless sensor solutions into the cloud environment allows the formation of an efficient fog computing environment [27].

Chen [28] considered a food chain as a cyberphysical system and proposed an intelligent approach for food traceability using the concept of fog computing. The author proposed an architecture that can handle the dynamics involved in food traceability. Luan et al. [29] defined the credibility of fog computing in bridging the gap between the mobile applications and the cloud computing. The authors emphasized the virtual resource utilization and location-based

service allocation as important aspects in building fog environments.

Okay and Ozdemir [35] defined a fog computing model for smart grids. The authors proposed a model which acts as a pivot between the cloud environment and smart grids. Their work focused on laying down the key points required in the formation of sustainable fog architecture, such as latency, self-healing, adaptability, security, and proximity. Tang et al. [36] developed a hierarchical architecture for analyses of big data systems in smart cities. Their architecture aimed at combining the multiple components of smart cities together to perform experimental evaluations of the collected data using event-driven systems. The existing solutions are application specific implementation of the sustainable fog computing having a limited scope in scalability for generic implementations.

Apart from the above approaches, utilizing service as a component of fog computing is an important paradigm in defining reliability and sustainability. Al Faruque and Vatanparvar [37] presented energy management as a service over fog computing. Energy efficient approaches can provide a stable solution for managing the services across the network. However, dependency only on the energy as a paradigm may allow the network to operate for a longer duration but cannot guarantee continuity in the case of threats and node failures. Further, web applications can be improved by provisioning of intelligent and efficient fog environment as well as smart gateways by utilizing the edge cloud architecture over fogging [38, 39].

2.3. Train Networks. Train networks are predefined and periodically configured networks which relay data utilizing the access points on the stations and antennas over the trains [40]. Trains allow support for heavy traffic as large equipment can easily be deployed over them [41, 42]. With a fixed route and path, a periodical approach can help to sustain the connectivity over the train networks. However, speed and handovers are the keys constraints for utilizing the train networks for crucial data-sustaining applications [43–45].

Train network provides a sustainable topology which does not change very often and the route of the trains is changed occasionally [46]. Such property allows ease of governance over networks. Further, with a predefined movement, it is easier to localize the servers placed on trains, which provides a controlled facility movement across the entire network. Such key aspects make train networks suitable for mobile server applications.

Trains can be used as a pivot for placing servers which can provide the facility of fog computing on the move [47]. Vehicle-based fog computing can be readily applied to the train network since this provides better stability and topology control over the entire network [48]. Apart from the advantages of using train networks, it is important to fix a location of the off-site server which will interact with the train servers for connectivity. A central or distributed control authority is also required which can keep a track of server activities as well as the alterations in the topology of the train. Such servers are the intermediate access points in connecting the train servers to the outer network.

3. Motivation and Problem Statement

The information and communication technologies have seen a tremendous growth in the number of users over the last decade. With an exponential increase in the number of users across different platforms, continuity of services and provisioning of quality is of utmost importance. Connectivity between almost all the devices over the network demands service providers to facilitate fast and efficient data processing. More users generate a large amount of data for transmission over the internet, which causes a huge overhead. In the recent years, a solution to such problem is provided in the form of fog computing, which aims at the formation of private and personalized cloud near the user, which decreases the latency involved in the data-sharing over the internet. Although it can provide an efficient solution for latency, it cannot help in sustaining the continuously increasing demand of users. Amendments are required either in the entire network layout or in the data-handling strategies, which can provide a scalable and sustainable approach for connecting a large number of IoT devices with low complexity.

The problem deals with the enhancement of connectivity between the IoT devices along with the distribution of load appropriately with an aim of connectivity to all. The novel network architecture and service allocation approaches are required which can sustain a load of increasing number of network devices without failure and can handle the pressure of device failure during network attacks.

4. Proposed Approach

The proposed approach aims at the formation of an intelligent and sustainable architecture for IoT devices using multiple UAVs and train networks. The train networks form the key part of the proposed approach by serving as the 5G-enabled mobile fog servers. The concept of fogging is taken to another level by forming a private near user cloud system rather than the traditional static fog servers. This helps in maintaining the logistics of fogging and allows attaining flexibility in covering a large number of devices with efficient load balancing. Train networks already exist in literature, where trains serve as the mobile terminals in between the user layer and the eNodeB considering a 4G-enabled network. However, the existing train network does not consider the near user fogging as well as cloud formation to improve the connectivity which can help in sustaining the pressure of increasing incoming requests as well as network failures.

4.1. Hybrid Fogging Using UAVs and Train Networks. This section presents the details on the network architecture comprising UAVs and trains. The train network is used as a location site for placing the heavy payloads, that is, fog servers. The fog servers are the key part of the near user cloud systems, which allows data processing within the communication zone of a user rather than transmitting it over the internet. The mobile servers reduce the cost involved in setting a new private cloud system, which otherwise remains static and required multiple connections for covering more users.

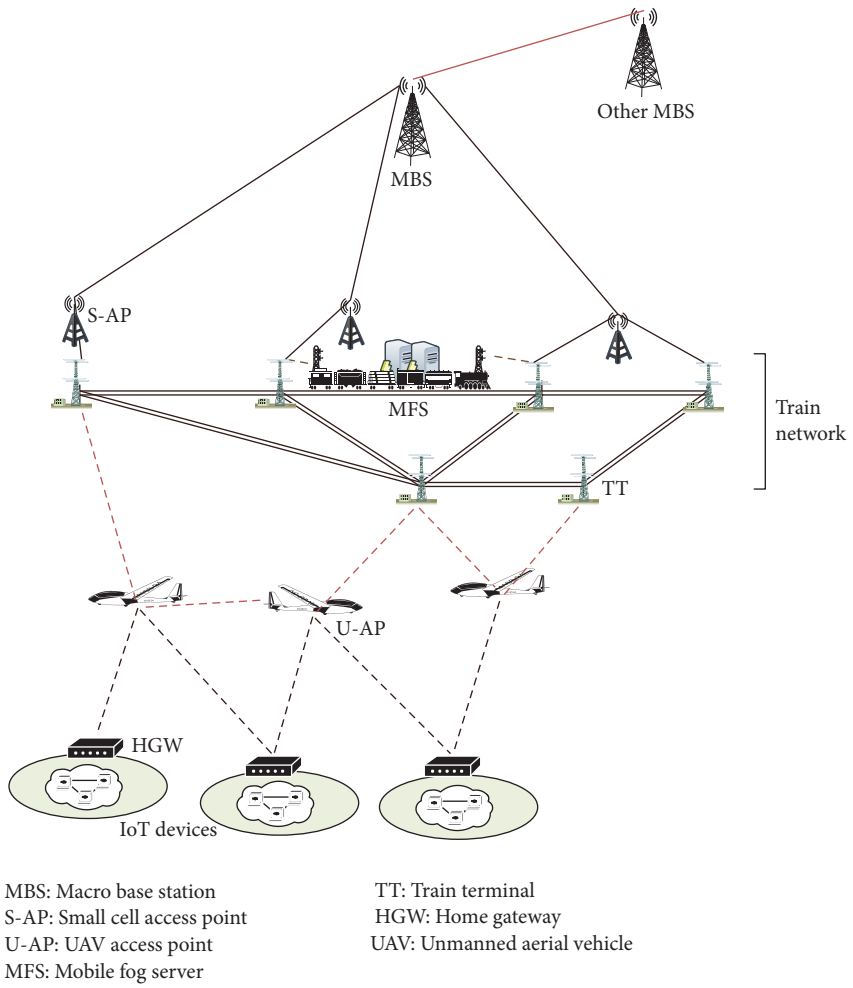


Figure 1: A representative illustration of hybrid fogging using train and UAV networks for sustainable IoT.

The mobile fogging approach allows fog servers to be traveling across the terminals with the maintenance of continuous links without consuming extra space. The storage on-the-go is the other key feature of mobile fogging. Further, the data can be easily distributed across the different trains depending on the terminals traversed. A query may arise for using UAVs as a direct fogging server between the small cell access point (S-AP) and the home gateways (HGW), which raises concerns regarding the payload supported by the aerial vehicles. Since fogging deals with the fast evaluations over data without transmitting it across the internet to a core public/private cloud, it requires heavy servers to be placed near user site, which is difficult for UAVs to accommodate. Thus, trains are used as a support for mobile fog servers.

In addition to this, a train system is a well-planned network, which seldom changes over the years with predecided and fixed route of each train. Considering all these aspects, a train system can provide strong support in the formation of sustainable mobile architectures. A representative illustration of the hybrid fogging using train and UAVs networks is shown in Figure 1 with a hierarchical view in Figure 2. The model comprises multiple macro base stations (MBS), each covering a zone which contains S-APs. The S-APs are the small cells which form the bridge between the train network and the MBS.

The underlying train network comprises two main components, namely, fog server (FS) which forms the key part of mobile fog cloud and train terminals (TT), which are the access points for connectivity between the FS and other network equipment. The direct communication with FS can also be considered; however, since the primary task of the proposed system is to form a sustainable communication setup, TT allows an extra layer of protection which helps in maintaining the continuous connectivity in the network. The connectivity between the HGW which are connected to multiple IoT devices is provided via an additional layer of UAVs which serves as UAV access points (U-AP).

An existing layer of femtocell can be used for connectivity between the HGW and the TT, but, for new network layouts, it is recommended to use dynamic UAVs as these aerial vehicles can reduce the cost involved in the implementation of static networks. Since the distance between the train network and HGW is less, UAVs can be used to provide wireless connectivity. This network can be operated over high-frequency wave system, but this would require low flying support from aerial vehicles as well as a dense network

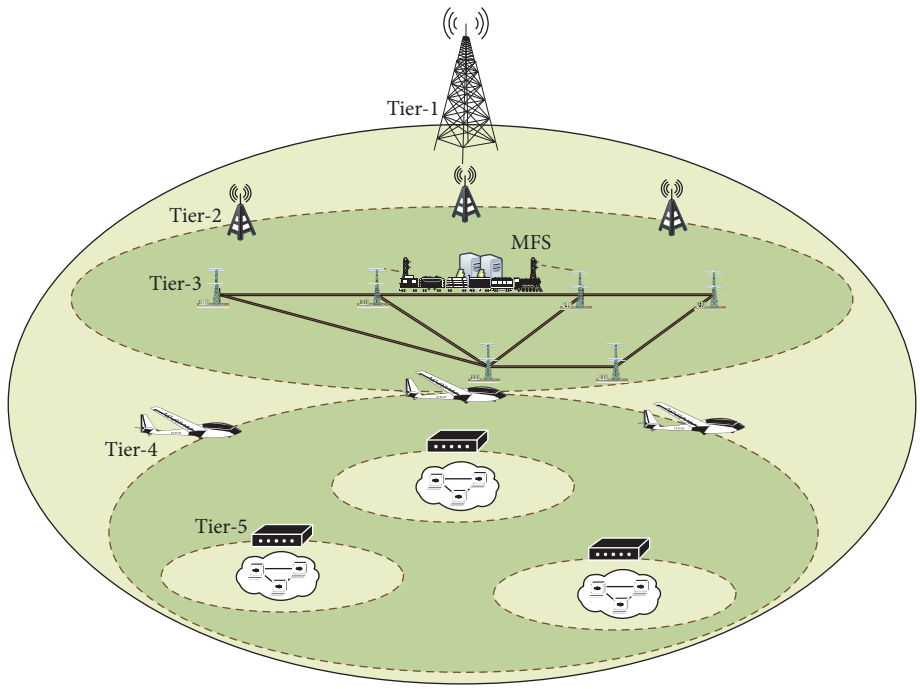


Figure 2: A hierarchical representation of hybrid fogging using train and UAV networks.

formation between them as high-frequency systems have a limited wavelength, which can be overcome by enhancing power and antenna characteristics.

4.2. System Model. The network model presented in the above section is modeled to satisfy the criteria for reliable and sustainable communications. Let B be the set of S-APs which connects the underlying components to the MBS. The underlying components include mobile FS and UAV layer which is the bridge between the HGW and the fog servers.

Let Z be the set of station terminals (TT) which are the transceiver antennas for connecting FS to the main network line. Let R be the set of trains and let F be the set of fog servers on each train. Let U be the set of UAVs which replaced the traditional femtocell of the networks, and let X be the set of users or IoT devices making continuous requests for services over the network.

The network aims at the formation of a dynamic graph $G(V, E)$ which updates after a certain interval when the nodes go beyond the transmission range or there is a change in the network topology due to node/link failure. Here, V is the set of vertices comprising network components and E is the set of edges representing transmission link between the nodes.

The link failure refers to the nonavailability of a route between the nodes despite the nodes being active, whereas the node failure refers to the nonavailability of nodes for intermediate relaying. The selection of the nodes for transmission is carried on the basis of reliability score R_s, sustainability S_t, and the output from a distributed node management (DNM) system which uses Wald's maximum model to optimize the connectivity between the nodes.

In the considered network, the reliability considering the graph $G = (V, E)$, where $V \in \{B \cup Z \cup R \cup F \cup U \cup X\}$, is given using [49] as

$$R_s = \sum_{i=1}^{n} G_i' P^i (1 - P)^{n-i}, \quad n = |V|, \tag{1}$$

where G_i' is the number of subgraphs with exactly i number of nodes when the nodes fail with a probability $Q = 1 - P$ and P is the probability of nodes without failure.

Thus, reliability can be defined as the probability of the existence of a route to a node in the set V during all time of connectivity. This means that a network can be reliable if there exists a graph G comprising the nodes x and y as a source and destination, respectively. Reliability can be considered as the only measure for sustainable networking as done by most of the solutions, but this can lead to inappropriate network formations because there can be a network with $R_s = \max$ that does not contain desired x and y. Hence, it is important to consider the reliability over links along with the reliability over nodes.

Thus, considering the reliability of connections, (1) is altered using [6], such that

$$R_s = \sum_{i=1}^{|E|} G_{x,y}' P^i (1 - P)^{m-i}, \quad m = |E|, \tag{2}$$

where $G_{x,y}'$ are the subgraphs containing x and y as vertices. Now, the probability of nonfailed nodes and graph formation

P is given as the ratio of available components for connections to the total number of components, such that

$$P = \frac{1}{|L|}\sum_{i=1}^{|L|}\left(\frac{C_a}{C_m}\right)_i, \quad (C_a)_i \neq 0. \tag{3}$$

Here, $\sum_{i=1}^{|L|}(C_a)_i = (|B'|+|Z'|+|R'|+|F'|+|U'|+|X'|)$ such that $|B'| \neq 0$, $|Z'| \neq 0$, $|R'| \neq 0$, $|F'| \neq 0$, $|U'| \neq 0$, and $|X'| \neq 0$, where every entity defines the set of available components on the particular layer. C_m is the number of components on all layers in set L such that $\sum_{i=1}^{|L|}(C_m)_i = (|B|+|Z|+|R|+|F|+|U|+|X|)$ and $|L| = 6$ as there has to be at least one vertex from all the six possible components. If any of these considerations is unsatisfied, the network fails.

Sustainability is defined in terms of connectivity over the graph between the nodes such that despite the number of failures there is always a route between the source and the destination. Out of the available connections, how many actually provides route defines the sustainability of the network; that is, if the network is active and a node guarantees connectivity in terms of R_s, it cannot guarantee sustainability until or unless it supports communication between the nodes.

Consider a scenario where the UAVs are overoccupied with the load; now the network is reliable, since there exists a path to support existing communications, but, to ensure further connections, the overheads induced due to a sudden increase in the services must be handled immediately to provide sustainable connectivity. Thus, the network sustainability is calculated as the ratio of free links to the available number of links; that is,

$$S_t = \frac{1}{|L|}\sum_{i=1}^{|L|}\left(\frac{C_b}{L_a}\right)_i, \quad (C_b)_i \neq 0, \tag{4}$$

where $\sum_{i=1}^{|L|}(C_b)_i = (|B''| + |Z''| + |R''| + |F''| + |U''| + |X''|)$, such that every entity defines the free links on each layer and L_a are the total links supported at each layer. The value 0 for free components refers to either a link or a node failure.

A graph can be reliable if it ensures the presence of end nodes in the subgraphs and it can be sustainable if it ensures the presence of a link between the end nodes of the subgraphs. The link stability can be attained by minimizing the interference between the nodes operating over the same spectrum as well as by keeping a minimum distance between the UAVs and TTs.

4.3. Decision Modeling and Optimization Problem. A decision system is formed over the reliability and sustainability of the network which helps to sustain the connectivity for longer duration without falling prey to a node or link failures. The decision system uses Wald's maximum model [18, 50] which is used in the case of involvement of two players that participate in a decision-making strategy in a sequential order. This model differs from other decision-making models by the case that the second player always knows the decision taken by the first player. This model fits well to the situation considered in this paper. The model can be applied either as a maximin formulation or as a minimax formulation.

In the considered model, the focus is on the reliability, sustainability, and the degree of connectivity. If $W\ (= T_r/T_p)$ is the weight assigned to the requests generated by the network, then the decision system aims at finding a state h in the network such that

$$h = \min_{x,y\in V} \max\left(\frac{T_p - T_r}{T_p}\right), \tag{5}$$

which refers to minimizing the maximum difference between the links available for handling the pending requests and the demanded links. Here, T_r is the number of links remaining and T_p is the total demanded links. (5) holds when the ideal state considers allocating single link to every service request. However, for load balancing, multiple links are utilized to distribute the load across the network; thus, (5) deduces to

$$h = \min_{x,y\in V} \max\left(k - \frac{T_r}{T_p}\right), \tag{6}$$

where k is the number of links fixed by the ideal state. The model can also be applied to deviation in the case if a decision is to be taken on the basis of multiple instances of subgraphs that are available over the same network. In such scenario, (6) can be represented as minimizing the maximum deviation between the states; that is,

$$h = \min_{x,y\in V} \max\left(\sqrt{\frac{1}{g}\sum_{i=1}^{g}\left(W_i - \overline{W}\right)^2}\right), \tag{7}$$

where g is the number of instances of subgraphs and $\overline{W}$ is the mean weight. Instead of mean weight, an ideal value can also be calculated to support continuous connectivity and Wald's model can be applied with respect to the ideal weight.

The entire network is subjected to three major paradigms which on successful optimization can provide highly balanced and sustainable computing for handling IoT devices. Out of the three optimization problems, one is given in (7) and the other two are as follows:

$$\begin{aligned} &\max \min\left(R_s\right), \\ &\max \min\left(S_t\right). \end{aligned} \tag{8}$$

Equation (8) aims at maximizing the minimum reliability and sustainability of the network.

4.4. Self-Aware and Sustainable Communication in IoT. The proposed architecture involves the hybridization of mobile nodes to provide a sustainable network which can guarantee a reliable and efficient communication over the IoT. The IoT devices require data evaluations to be performed at a rapid pace so as to enhance the reply time to the query maker. The data evaluations depend on the structuring of data, which is not in the scope of this paper; but the pace depends on the type of network and location of the server to perform evaluations, which is considered in this paper.

With the concept of fog computing, the near user site cloud formation provides extensive support for storage,

retrieval, caching, and mining of information without any latency. However, the cost and the periodicity of user requests affect the performance of the existing static fog computing. Thus, the proposed SACA utilizes a mobile infrastructure to sustain as well as grow the processing and computation power of a network. This allows handling of a large number of users even in the scenario of network threats, attacks, and node failures.

The proposed SACA utilizes the unique sensor feeding, sensor signatures, and knowledge-depth graphs for the formation of a sustainable architecture, which keeps a track of network states and helps in allocating the server on the basis of load and network alliance. This strategy allows the selection of efficient links and allows load migrations in the case of urgency or operational issues.

4.4.1. Sensor Feeding and Signatures. The proposed SACA utilizes the existing sensor features to find optimality in the network which can guarantee optimization over reliability, sustainability, and deviation issues. The network comprises IoT devices which have a network card installed providing a unique signature to every device. In the proposed approach, the registration is done in two ways to attain reliability. Every device which registers itself for the connectivity over the proposed model is given a unique registration number by the corresponding FS; however, FS is unaware of the registration sequence and the device to which it is allocated. This process is termed as the sensor feeding.

In sensor feeding, each device in the network requiring connections makes a request to its HGW, which keeps a track of its physical address and gathers all the information about the device activity including the type of data it operates on and allocates a unique sequence counter. Now, as soon as HGW maintains a list of incoming devices, it demands registration IDs from the FS via intermediate access points (UAVs). It is to be noted that this paper does not consider any intelligent activity on the UAVs and treat them only as a forwarding router. FS gives a set of registration IDs to an HGW and also shares the IDs across the other FSs, which helps to maintain a connection on the move. The HGW allocates the registration IDs randomly to every connected device, thus maintaining an abstraction of the sensor signatures from the FSs.

After an initial agreement between the IoT device, HGW, and FS, the HGW sends the property list to FS which now knows the type of data it will receive for the particular registration ID but is unaware of the device making the request. On receiving, FS acknowledges if it can provide the requested services; or, otherwise, it sends the request to the DNM which is placed at the station terminal. Thus, the TT not only acts as train access points but also has a capability of deciding another FS which can handle a service request that is initially declined by an initial FS. The same procedure of sensor signatures and feeding is given in Figure 3.

DNM is invoked only if an FS is unable to handle the service request; otherwise, the network operations continue without involving new network operations. This helps in maintaining lower latency by reducing the operational impact of DNM. However, in a highly overloaded condition, DNM proves to be handy as it helps to take a predecision and allows efficient allocation of service requests to the available servers. Contrary to this, invoking DNM on every service request increases the handling overhead; thus, the emphasis is given on the policy of invoking DNM only when required.

4.4.2. Knowledge-Depth Graphs. The DNM forms an integral part of the network and is invoked in the absence of service validation from a requested FS. Usually, the DNM allocates services to FS randomly from the point of view of just handling them. However, it also keeps a record for each state of the network and handles the situation when an FS declines to handle the request despite availability.

The DNM utilizes the concept of Knowledge-Depth (KD) Graphs. The KD graphs are formed by the union of two dynamic graphs, one with knowledge as the property assigned to each of its vertex and the other with the depth of knowledge as a property. The knowledge graphs are represented as $G_1^* = (V, E, K_g)$, where K_g is the knowledge set for the vertices in V. Value of each element in K_g is calculated as the ratio of total degree of the node (D_t) to the total edges in the network; that is,

$$K_{g,i} = \frac{D_{t,i}}{|E|}, \quad i \in V. \tag{9}$$

Higher value for the degree of a node represents better knowledge of the network; similarly, the depth graphs are defined as the level of knowledge, which is expressed as the ratio of the sum of direct links (D_l) in all the subgraphs containing the node to the total links available on the layer/tier to which the node belongs, such that

$$K_{d,i} = \frac{D_{l,i}}{L_{a,i}} \tag{10}$$

and $G_2^* = (V, E, K_d)$. Thus, for each set of vertices and edges, there are two graphs available which are termed as the KD graphs. These graphs help in taking a decision on the basis of requirement of the load. A depth graph is used when the load is to be transferred across the nodes of layers other than FS, whereas knowledge graph is utilized when the load is to be managed across the FS. However, the network can operate using a single KD graph by generalizing the weight associated with the vertices such that the optimal graph is given as $G_f^* = (V, E, K_w)$, and

$$K_{w,i} = \eta_1 K_{g,i} + \eta_2 K_{d,i}, \tag{11}$$

where η_1 and η_2 are the balancing constants for managing a relation between knowledge and depth such that $0 \leq \eta_1 \leq 1$ and $\eta_1 \leq \eta_2 \leq 1$ as depth is more important when the knowledge is available.

4.4.3. Load-Based Server Allocation. The proposed SACA aims at the formation of a sustainable network which guarantees connectivity even in the case of node/link failures. The proposed approach utilizes the KD graphs to take a decision on the basis of network load. The DNM takes a decision on allocating the server on the basis of trivial approach by

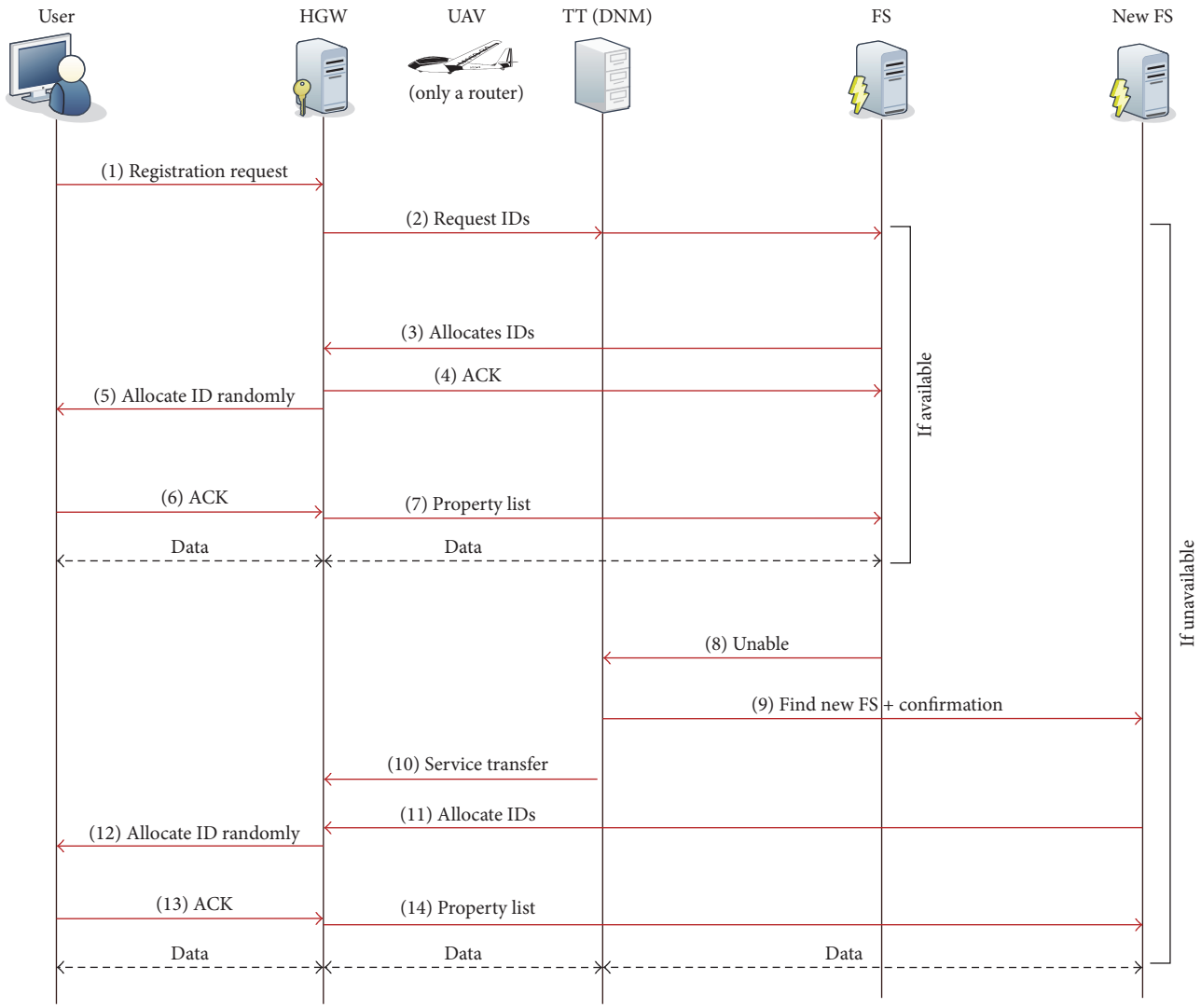


Figure 3: An illustration of sensor feeding and signature registration procedure.

checking the current load of every available server and the KD properties.

The KD graph formed is used to check the available servers for hosting the requested services by an HGW via UAVs and TTs. DNM can take the decision either on the basis of knowledge graph or depth graph or by using the common weight graph by utilizing (9)–(11). DNM is also capable of selecting multiple servers for handling the data from the same source by dividing the operations between the multiple servers. In the case of availability of multiple servers with similar load handling capabilities, the one with better KD graph properties is selected.

4.4.4. State Maintenance and Learning. Continuous connectivity depends on the state maintenance and learning about the network situation for maintaining reliability and sustainability. The state maintenance is performed by defining the order of connection. Learning can be achieved only for the first-order connections, since the nodes can have an exact status of the other node. A detailed overview of the state dependency diagram for learning is shown in Figure 4. The dependency diagram is formed by considering the connectivity between the different tiers of the network. For example, the nodes with direct connectivity are given first-order dependency and nodes with an intermediate are given second-order dependency and so on. This allows easy learning mechanisms and maintenance of "who is connected to whom."

The state maintenance and learning depend on the DNM which play a pivotal role in handling transmission across the entire network. The depth of connectivity defines the learning mechanism of a network. The utilization of mobile FS in handling large processing and storage requests is supported by the formation of a learning system which can be updated with low complexity.

Whenever a selection operation is performed across the FS, the DNM maintains a log on the basis of inputs received from the handling FS. The FS manages the traffic and provides support to DNM for watching the trend in the upcoming

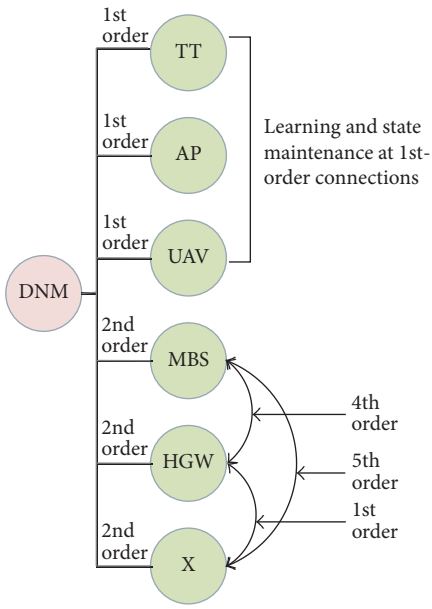


Figure 4: An illustration of state dependency for learning.

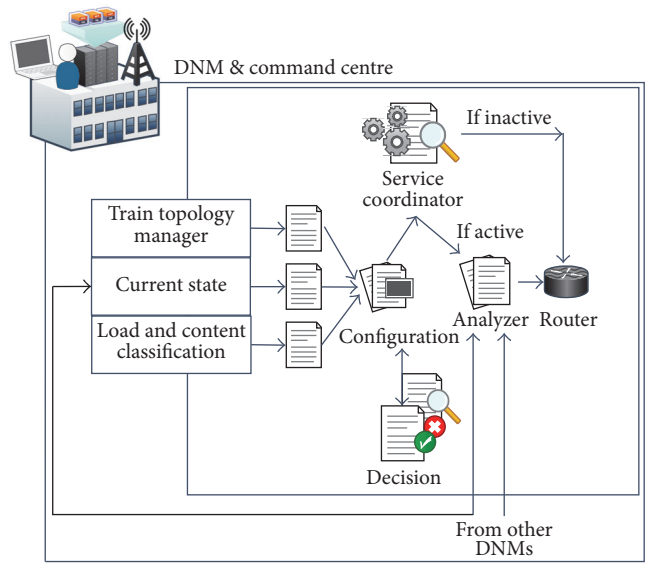


Figure 5: An illustration of DNM deployment as a spy model.

traffic and taking a decision for regulating the role of nodes involved as a fist-order connection as well as the selection of new routes in the case of failures.

The DNM and FS are equipped with the feature of sending warning signals across the entire network beforehand so as to prevent any network threat as well as an anomaly. A remedy to threat is made by not considering the server or node for further communication until the problematic server confirms positive role by sharing its consistent logs without participating in the communication.

4.4.5. Spy-Based Deployment. The entire network is laid as the initial architecture defined comprising train and UAV network. The operations are performed as a regular network except for the fact that the FS plays a key role in providing near user site cloud services for handling a large number of service requests with low latency. An intelligent system is formed over the TTs, which is termed as DNM which manages and controls the entire network by managing the network state, node configurations, and state logs. An illustration of spy-based deployment of various features over a TT-DNM is shown in Figure 5.

The deployment as a spy allows close control over the FS and a virtual control over the entire network. The train topology, current network state, and traffic controller form the key part of DNM. All these are passed as an input file to form the main configuration file. This file simply maps the network state to the train topology and the traffic condition by following the time as a controlling metric. Then, a configuration analyzer is invoked which takes a decision on the correctness of network states, after which a service coordinator takes a call on selecting the appropriate server for handling the user requests.

An intermediate analyzer is also provided which takes input from the service coordinator and the inputs from other DNMs to maintain a list of available FS. An API support is provided on the DNM server to easily manage and configure the network policies and maintain a connectivity state. The service coordinator is responsible for managing the information flow across the network. The learning is provided by maintaining file logs of each state which are then passed to the current state analyzer.

4.4.6. Network Alliance for Failures Detection. The network alliance is a node cooperative system managed by DNMs for preventing network against induced failures which may or may not be caused due to a vulnerability or network attack. The network alliance aims at handling network in the case of node/link failures without affecting its performance. The procedure for network alliance is simple and depends on the state dependency model, virtual DNM control, and periodic analyses of the sustainability value.

DNM periodically shifts the control to one of the nodes of every tier which operates as a virtual DNM. All the nodes in the tiers register and share their calculated sustainability value with the virtual DNM node. The virtual DNM node then provides all the accessed information to the TT-DNM which matches the attained information with its state diagram. Such application allows DNM to possess a virtual control over the entire network.

The procedure of network alliance can be conducted either periodically or during failure in the network. The value of R_s and S_t can also be used to decide on conducting network alliance procedures. The steps for network alliance are presented in Algorithm 1. The network alliance helps in understanding the failures in the network, which allows taking a decision on changing interaction procedures with the nodes so as to reduce the delay.

Lemma 1. *The depth, knowledge, and probability of connection between the nodes increase with a higher degree of connectivity which improves the reliability of the network.*

(1) **Input**: Current State, DNM - S_t and R_s
(2) **Output**: Decision for continuity or change link
(3) set interval
(4) **while** Transmission Continues **do**
(5) check for interval
(6) request periodic update
(7) select most communicating node from each layer
(8) send virtual control to the node
(9) receive and map state dependencies
(10) **if** ambiguous $\|S_t < \overline{S_t}\| \; R_s < \overline{R_s}$ **then**
(11) eliminate node from state-dependencies
(12) update neighbours
(13) **else**
(14) continue
(15) **end while**
(16) **end while**

Algorithm 1: Network alliance for prevention against failures.

Proof. From (9)–(11), the knowledge and depth increase affecting the connectivity between the nodes; this connectivity directly affects the probability of connections between the nodes as the number of subgraphs containing the source and destination will increase with an increase in the overall degree of the nodes. Using (2), with the increase in probability and number of subgraphs, the reliability of the network increases. □

Lemma 2. *With reliability attaining a maximum value, the sustainability increases and maximizes if available degree per node is equal to k times the remaining links* (T_r).

Proof. Reliability of a network can maximize with an increase in the number of subgraphs containing the source and destination nodes, which further increases with an increase in the probability of connectivity. However, a reliable network does not always guarantee sustainable formations, which depends on the number of connections available and supported by each node. From (5), if $T_p = kT_r$, maximum number of links are available, which increases the number of free links (see (4)), thus, maximizing the sustainability. □

Remark 3. Reliability and sustainability can simultaneously maximize their minimum value when $P = 1$.

Proof. With a higher degree per node, a sufficient number of subgraphs are available for connectivity between the nodes, which makes $P = 1$ and, from (1), R_s attains maximum. Now, with R_s at maximum and $P = 1$, $C_b = L_a$, which makes S_t attain a maximum value. □

Remark 4. Traffic variations and an increase in the number of users making connection demands affect the network reliability.

Proof. With an increase in the number of users making continuous service requests, the probability of falling in different subgraph increases as all the users may not fall in the same subgraph; this decreases the reliability of the network, and alternative paths are required for transmissions. This is the condition for the requirement of load balancing. □

Remark 5. The minimum number of connections required to sustain communication is greater than or equal to k, where k defines the minimum rule for connectivity. The number of station terminals required to allow this transmission depends on the rule of $2r + Q$, where r is the radio range of terminals and Q is the length of a train.

Proof. From (5) and (6), $T_p = kT_r$, which defines the condition for minimum number of connections as stated by the lemma. Now, for the number of terminals, the trivial rule of $2r + Q$ is followed as a measure of distance between two TTs, as shown in Figure 6. Considering wireless multihop transmissions, the antennas are placed on the both ends of train which are connected to each other via FS; now, the antennas over train and TT are assumed to have common radio range r, which means the maximum gap between two TTs can be maximum up to $2r+Q$. The placement of TTs in the proposed architecture should be governed by this rule. □

5. Performance Evaluation

The proposed SACA is evaluated in three parts. The first part presents the numerical analyses for reliability and sustainability, the second part presents evaluation of SACA in simulation environment, and the third part evaluates SACA for its sustenance in the presence of intruders and attackers resulting into node failures.

5.1. Numerical Analyses. In the numerical analyses, the results are presented for the variation of a variable (connections) and its impact over the network reliability and sustainability. The numerical analyses are conducted using Matlab™ with configurations given in Table 1.

A total of 1000 users made consistent connection demand in a network operating with 5 tiers. The results are recorded

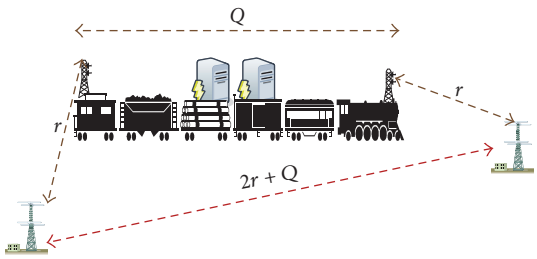


Figure 6: An illustration of maximum distance sustainable between two TTs.

Table 1: Parameter configurations for numerical analyses.

Parameter	Value	Description
$\|L\|$	5	Number of tiers
$\|X\|$	1000	Number of IoT devices
$\|B\|$	2	Access points
$\|Z\|$	5–20	Station terminals
$\|R\|$	4	Number of trains
$\|F\|$	4	Number of fog servers
$\|U\|$	5	Number of UAVs
k	2	Number of connections per node
G	Random	Graph for analyses
η_1, η_2	0.5	Balancing constants

for variation in the number of TTs with respect to the connections demanded by the nodes. The variation in the probability of connectivity affects the performance of the network. With a larger number of free connections, the network capability in handling more users increases. Figure 7 presents the results for variation in the network reliability with variation in the number of terminals available for connectivity operating with 5 UAVs. Also, the graph includes the impact of variation in the number of subgraphs including the source and the destination. With a higher value of intermediate nodes and terminals for connectivity, more links are available for connections. Also, the increase in the number of links is accompanied by an increase in the alternative routes between the source and destination which increases the probability of connectivity of the overall network, thus resulting in an increase in the overall reliability of the network.

A reliable network may or may not provide sustainable connectivity as it may have a connection for one set of nodes, while on the other hand it may not provide connectivity between the requested nodes. Since the numerical simulations are performed in a common graph, network sustainability attained a higher value with an increase in the number of TTs as shown in Figure 8. The increase is marked by an increase in the overall probability of connectivity for the users making continuous requests. Thus, it is concluded from the numerical analyses that the number of intermediate nodes and the connection supported by them heavily impact the performance of the network. However, deployment of more number of TTs and other intermediate nodes will increase the overall cost of network. Thus, selection of an optimal value for the number of intermediate nodes can be

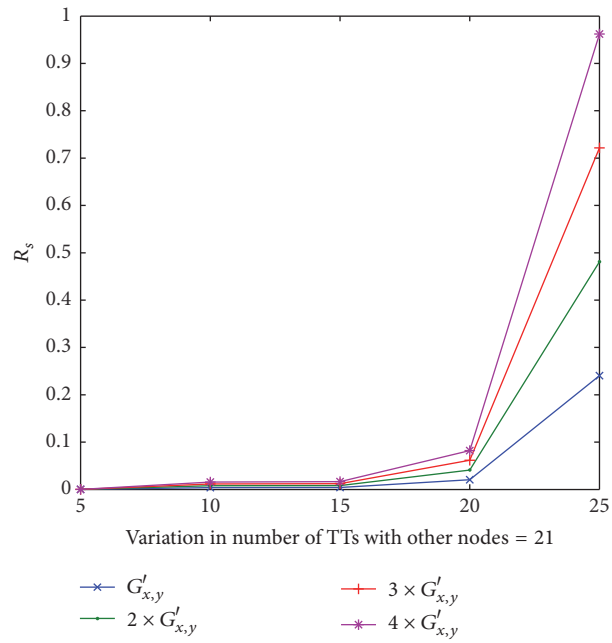


Figure 7: R_s versus variation in the number of TTs and subgraphs containing source and destination.

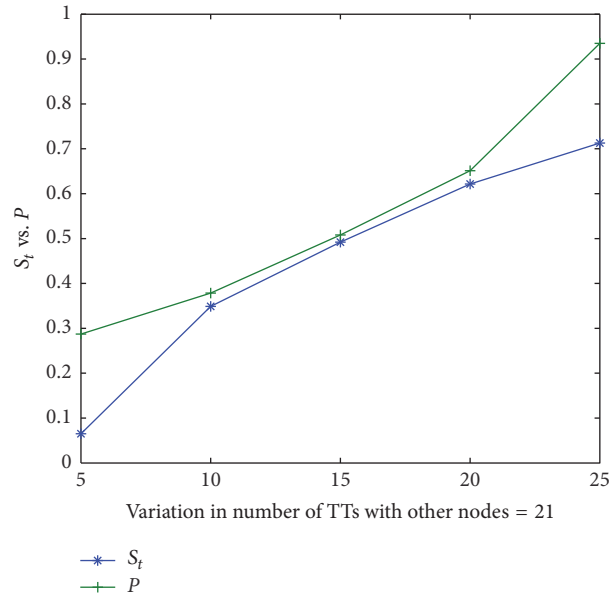


Figure 8: S_t and probability of connectivity versus variation in the number of TTs.

performed by considering the level of stability which should be sufficient enough to maintain continuous connections.

5.2. Network Simulation Analyses. The network simulations are conducted using Matlab by creating a scenario comprising all the components as network nodes operating using the configurations of a wireless network. The range of each node is kept fixed at 500 m. The length of a train is taken to be 50 m. The number of aerial nodes varied between 5 and 20

Table 2: Parameter configurations for simulations.

Parameter	Value	Description
$\|L\|$	5	Number of tiers
$\|X\|$	200–1000	Number of IoT devices
$\|B\|$	50–100	Access points
$\|Z\|$	20–50	Station terminals
$\|R\|$	5	Number of trains
$\|F\|$	5	Number of fog servers
$\|U\|$	5–20	Number of UAVs
k	2	Number of connections per node
Mobility-IoT	Random waypoint	Mobility model ground
Mobility-UAVs	Cooperative mode	Mobility model aerial
UAV speed	50 kmph	UAV movement
Train speed	150 kmph	Train movement
r	500 m	Radio range
Q	50 m	Length of train
Runs	20	Simulation runs
Traffic type	CBR	Traffic over TCP
Packet size	1024 bytes	Average packet size
Buffer	5000–20000	Buffer capacity
Interval	2 p/s	Time to wait before sending
Initial rate	256 kbps	Initial transmission rate

and the traffic is created using Poisson distribution. A total of 20 simulation runs are traced for analyzing the performance of the proposed SACA in terms of end to end delay and packet loss.

A variation in the number of users is considered with the intermediate links in dynamic state operating in an environment with a failure rate of 10% and 20%. The failures in simulations are dynamically induced over the random links. During simulations, the source and destination are kept at a distance having at least 2 intermediate hops from the failed nodes. Other configuration details for simulations are shown in Table 2.

The IoT devices are modeled using random waypoint, whereas cooperative framework [51] is used for the aerial nodes. In the performed simulations, five trains are made to run with fixed periodicity at a speed of 150 kmph. The IoT devices operate in "Request" mode, which means every device in the network demands authority for transmission during simulations. The baseline of the proposed approach is defined with no failures and complete connectivity between the nodes. Link state routing is applied using K_w as the weight metric. The simulation results are evaluated for the end to end delays and packet loss. The end to end delay ($E2D$) is calculated using [52] as

$$E2D = D_{\text{transmission}} + D_{\text{propagation}} + D_{\text{queue}} + D_{\text{processing}}, \tag{12}$$

where $D_{\text{transmission}}$ is calculated as the ratio of number of bits transferred to the link speed (rate of transmission), $D_{\text{propagation}}$ is the ratio of distance between the nodes to the channel speed, D_{queue} is the waiting time of packets before the beginning of processing, and $D_{\text{processing}}$ is the delay induced during the forwarding of packets. The packet loss is calculated as the ratio of lost packets to the total transmitted over the network.

With a variation in the failure of nodes from 10% to 20%, the end to end delay is recorded 45.7% and 48% higher than the baseline which observed a maximum end to end delay of 27 seconds. The higher values include the entire session delays as shown in Figure 9. With an increase in the number of users, more requests are made in the network, which affects the availability of connections per component. This increase is reflected in terms of average waiting time, which is included in the end to end delay graphs. However, focusing on the level of complexity considered in the simulations and the number of users making simultaneous requests over limited resources, the lesser delays justify the efficiency of the proposed model.

Further, with lesser values of delay, the overall performance of the proposed SACA is very high. The average packet loss for the entire session is very less as shown in Figure 10. The baseline operations recorded 20.8% and 34.2% lower packet loss in comparison with operations at 10% and 20% failure rate of nodes. The lowest value of 1.21% is recorded for packet loss in the proposed approach. However, with an increase in the number of users and constraints by the increase in average waiting time, the packet loss increases but does not go beyond a value which can affect the network. The overall delivery ratio of the proposed approach remains higher than 90% even in the scenarios of induced failures.

5.3. Sustainability Analyses. The network is sustainable if it provides strong connectivity support even in the case of

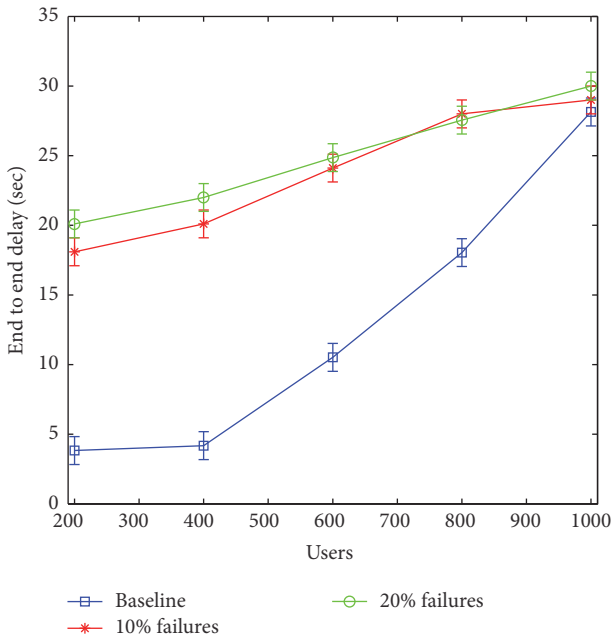


Figure 9: End to end delay versus users.

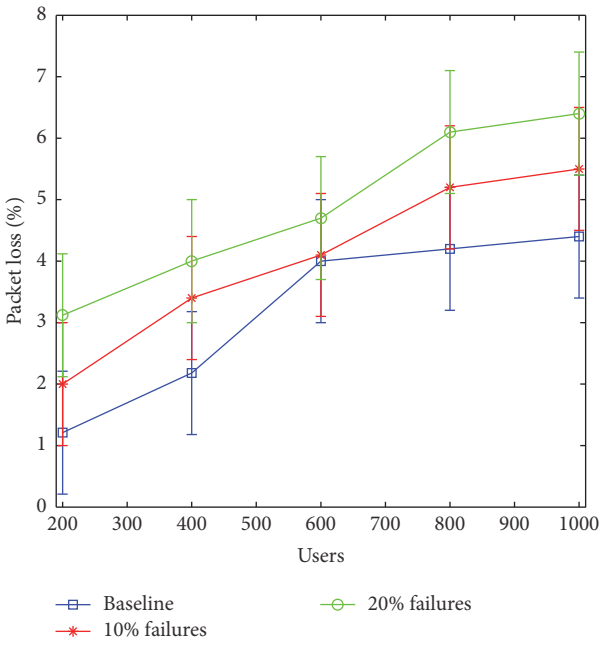


Figure 10: Packet loss (%) versus users.

node/link failures. The maximum cases of failures arise in a network when an attack is induced in the network. An attack can be as severe as resulting into the leakage of information, compromising of user accounts, or even the shutdown of the entire network.

Three different attack scenarios are considered for evaluation of the sustainability of the proposed SACA model, namely, Sybil attack, distributed denial of service (DDoS), and wormhole attack. Sybil attack is vulnerability caused when the users of a system induce false reputation for the intruder node [53]. A DDoS attack is caused by multiple

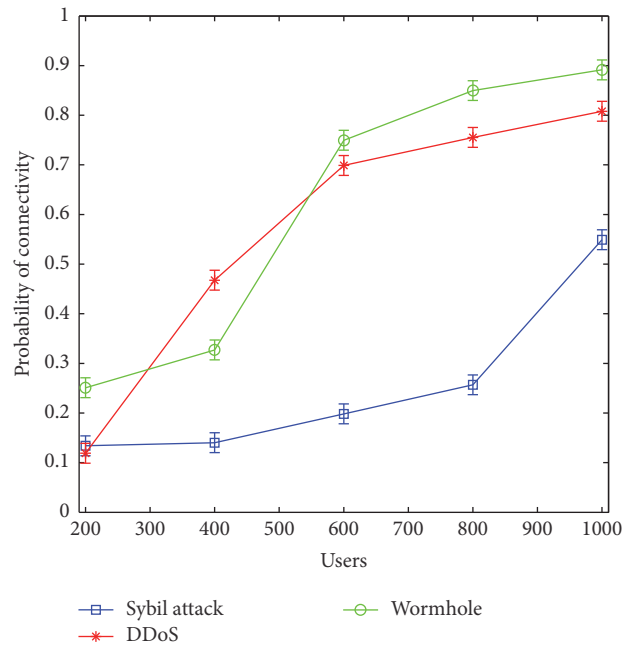


Figure 11: Probability of connectivity versus users.

flooding over a single path by network nodes affecting the traffic flow over a particular link [54, 55]. Wormhole attack is a type of false routing by making a glimpse of the presence of an alternative shorter route to the actual node [56]. All these attacks are highly critical for any type of network. These attacks are also time dependent as they take some time to penetrate into the entire network. The results shown in the paper are evaluated for each simulation cycle of 100 seconds.

The proposed approach is evaluated for its probability of connectivity and degree of sustainability in the presence of these attacks. A similar network as that used in the simulations is considered for the evaluation of proposed model in the presence of attacker nodes. The attacks are induced over 20% of the total nodes in the network. For Sybil attack, a false reputation index is given to the attacker nodes and the network pretends to consider these nodes as legitimate. Any data transmitted to these nodes is considered as a dropped packet and results are recorded. Similarly, for DDoS, 20% of the network nodes preoccupy the links leaving a lesser number of subgraphs with source and destination, and, for the wormhole, 20% of the nodes give similar next hop address for generating false routes.

Initially, the results are recorded for the probability of connectivity as shown in Figure 11. Sybil attack affected the proposed model more in comparison with the DDoS and wormhole attack. This is because of difficulty in identifying the nodes by the DNM while performing network alliance. The dependency over a false node for network alliance causes a reduction in the transmission rate as well as the probability of connectivity. However, with an increase in the number of nodes, more alternative paths are available, which allows an increase in the connectivity. The key advantage of the proposed approach is its high provisioning of sustainable networking. Despite the presence of attacker nodes in the

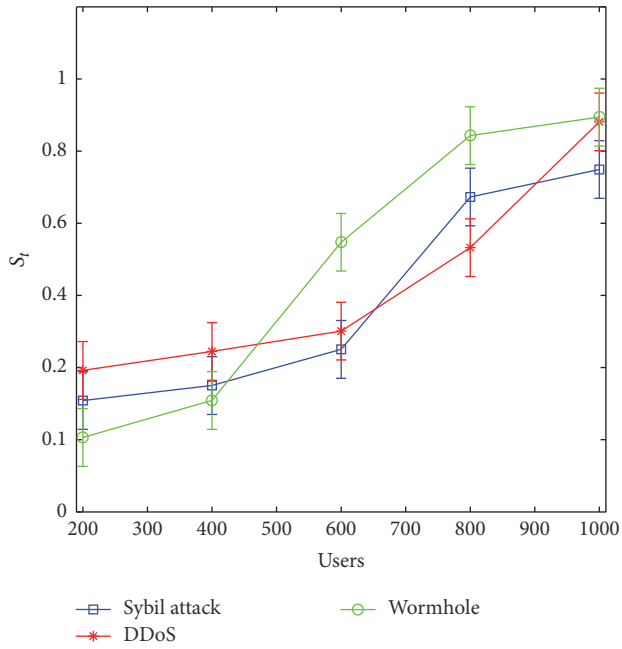


FIGURE 12: Sustainability (S_t) versus users.

network, the proposed model which utilizes a periodic concept of network alliance via DNM over TTs is capable of keeping aloof the vulnerable nodes from the selected path. Further, the quick succession of next hop in the case of node/link failure during an attack in the proposed model allows a high value for network sustainability as shown in Figure 12. With an increase in the number of users, the sustainability of the network further increases as the number of subgraphs containing the source and destination increases along with an increase in the number of available links for connectivity.

6. Discussions and Open Issues

The proposed SACA is capable of providing continuous services even in the adverse conditions which prevent data forwarding as well as decision-making on selecting next hop. SACA uses a DNM module which operates over one or all TTs of train network and allows efficient control over the network. The use of state diagram dependency allows resolution of conflicts involved in the selection on next hop and network alliance is used to handle the failures. The proposed approach is efficient in providing low delay and low packet loss transmission with high reliability, probability of connectivity, and sustainability as proven by the results.

There are other certain architectures which aim at provisioning of sustainable and robust connectivity over IoT devices but in a particular application scenario. Most of them use the existing underlying network model and do not present any variation in network formation. The existing solutions only provide a service-level solution for enhancing the connectivity for IoT devices. A state-of-the-art comparison is presented in Table 3 which presents key contributions for sustainable IoT along with their ideologies.

Despite the existing and proposed solution for sustainable IoT, there are several other issues which must be handled for the robust and reliable communication in IoT devices. These include the following:

(i) Handling energy constraint for reliable communications

(ii) Approaches for privacy preservation and fast authentication of IoT devices

(iii) Handling handovers efficiently to provide less latent architectures

(iv) Service divisibility and abstraction for context-aware IoT

(v) Self-organization and autonomous decision-making for IoT devices to handle unexpected failures and changes in the network

7. Conclusion

IoT demands high support from the underlying network. An efficient network can help in sustaining the services across the devices with prolonged connectivity. In this paper, the problem of sustainable and reliable flow of information is considered over a hybrid network formation. The solution proposed in this paper uses a concept of hybrid multimodular self-aware architecture which helps in providing fault-free communication. The proposed model uses a distributed node management (DNM) system which takes care of network connections and helps in identifying nodes which are reliable and can sustain the pressure of increasing demand of users. An optimization problem is formulated using Wald's maximum model. Analyses show that the proposed approach is capable of providing sustainable and reliable connectivity with lesser delay and fewer packet losses. The proposed approach is also capable of handling transmissions even in the scenarios that are under the threat of Sybil, wormhole, and DDoS attacks. By intelligent decision-making and self-awareness regarding the state of network components, the proposed model can guarantee connectivity even in unfavorable conditions.

In the future, we shall be aiming at extending the features of the proposed DNM to more realistic scenarios and testing it using hardware-assisted emulations.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (B0190-16-2032, Development of Operating System Security Core Technology for the Smart Lightweight IoT Devices) as well as by the Soonchunhyang University Research Fund (no. 20150690).

Table 3: State-of-the-art architectures for efficient connectivity in IoT.

Approach	Ideology	Sustainable	Key features	Application-specific	Secure	Network novelty
Cirani et al. [7]	Authorization service architecture	Yes	Constrained application protocol	No	Yes	No
Zgheib et al. [30]	Semantic data driven architecture for healthcare	Yes	Semantic Sensor networks	Yes	No	No
Gupta et al. [31]	Cloud centric architecture	Yes	XML web services	Yes	Yes	No
Barbosa et al. [32]	Architecture for emotional smartphones	No	Happy system architecture	Yes	No	No
Sarkar et al. [33]	Distributed architecture for IoT	No	Automated service management	No	Yes	No
Flauzac et al. [34]	SDN-based architecture	No	Distribution of security rules	No	Yes	Yes
Proposed SACA	Hybrid mobile fog servers	Yes	Distributed node management	No	Yes	Yes

References

[1] D. Bol, J. De Vos, F. Botman et al., "Green socs for a sustainable internet-of-things," in *Proceedings of the IEEE Faible Tension Faible Consommation (FTFC '13)*, pp. 1–4, IEEE, Paris, France, 2012.

[2] L. Fritsch, A.-K. Groven, and T. Schulz, "On the internet of things, trust is relative," in *Proceedings of the International Joint Conference on Ambient Intelligence*, pp. 267–273, Springer, Amsterdam, The Netherlands, November 2011.

[3] N. K. Giang, J. Im, D. Kim, M. Jung, and W. Kastner, "Integrating the epcis and building automation system into the internet of things: a lightweight and interoperable approach," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 6, no. 1, pp. 56–73, 2015.

[4] C. E. El Kaed, I. Khan, H. Hossayni, and P. Nappey, "Sqeniot: Semantic query engine for industrial internet-of-things gateways," in *Proceedings of the IEEE 3rd World Forum on Internet of Things (WF-IoT '16)*, pp. 204–209, Reston, Va, USA, December 2016.

[5] B. B. Snchez, D. S. de Rivera, and L. Snchez-Picot, "Building unobtrusive wearable devices: an ergonomic cybernetic glove," *Journal of Internet Services and Information Security (JISIS)*, vol. 6, pp. 37–52, 2016.

[6] T. Robles, R. Alcarria, D. Martín et al., "An iot based reference architecture for smart water management processes," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 6, no. 1, pp. 4–23, 2015.

[7] S. Cirani, M. Picone, P. Gonizzi, L. Veltri, and G. Ferrari, "IoT-OAS: an oauth-based authorization service architecture for secure services in IoT scenarios," *IEEE Sensors Journal*, vol. 15, no. 2, pp. 1224–1234, 2015.

[8] S. M. Ghaleb, S. Subramaniam, Z. A. Zukarnain, and A. Muhammed, "Mobility management for IoT: a survey," *Eurasip Journal on Wireless Communications and Networking*, vol. 2016, no. 1, article 165, 2016.

[9] G. Marques, N. Garcia, and N. Pombo, "A survey on IoT: architectures, elements, applications, QoS, platforms and security concepts," in *Advances in Mobile Cloud Computing and Big Data in the 5G Era*, vol. 22 of *Studies in Big Data*, pp. 115–130, Springer International Publishing, Cham, Switzerland, 2017.

[10] K. Kai, W. Cong, and L. Tao, "Fog computing for vehicular Ad-hoc networks: paradigms, scenarios, and issues," *Journal of China Universities of Posts and Telecommunications*, vol. 23, no. 2, pp. 56–96, 2016.

[11] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenw, and B. Koldehofe, "Mobile fog: a programming model for large-scale applications on the internet of things," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Mobile Cloud Computing*, pp. 15–20, ACM, Hong Kong, August 2013.

[12] V. Sharma and R. Kumar, "Teredo tunneling-based secure transmission between UAVs and ground ad hoc networks," *International Journal of Communication Systems*, 2016.

[13] V. Sharma, I. You, and R. Kumar, "Energy efficient data dissemination in multi-UAV coordinated wireless sensor networks," *Mobile Information Systems*, vol. 2016, Article ID 8475820, 13 pages, 2016.

[14] V. Sharma and R. Kumar, "Cooperative frameworks and network models for flying ad hoc networks: a survey," *Concurrency Computation: Practice and Experience*, 2016.

[15] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Drone small cells in the clouds: design, deployment and performance analysis," in *Proceedings of the 58th IEEE Global Communications Conference (GLOBECOM '15)*, pp. 1–6, December 2015.

[16] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Unmanned aerial vehicle with underlaid device-to-device communications: performance and tradeoffs," *IEEE Transactions on Wireless Communications*, vol. 15, no. 6, pp. 3949–3963, 2016.

[17] A. Merwaday and I. Guvenc, "UAV assisted heterogeneous networks for public safety communications," in *Proceedings of the IEEE Wireless Communications and Networking Conference Workshops (WCNCW '15)*, pp. 329–334, IEEE, New Orleans, La, USA, March 2015.

[18] A. Wald, "Statistical decision functions," in *Breakthroughs in Statistics*, Springer Series in Statistics, pp. 342–357, Springer, New York, NY, USA, 1992.

[19] D. Kyriazis, T. Varvarigou, D. White, A. Rossi, and J. Cooper, "Sustainable smart city IoT applications: heat and electricity management & Eco-conscious cruise control for public transportation," in *Proceedings of the IEEE 14th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM '13)*, pp. 1–5, June 2013.

[20] S. Wen, X. Zhu, X. Zhang, and D. Yang, "QoS-aware mode selection and resource allocation scheme for Device-to-Device (D2D) communication in cellular networks," in *Proceedings of the IEEE International Conference on Communications Workshops (ICC '13)*, pp. 101–105, IEEE, Budapest, Hungary, June 2013.

[21] S. Mayer, D. Guinard, and V. Trifa, "Searching in a web-based infrastructure for smart things," in *Proceedings of the 3rd International Conference on the Internet of Things (IOT '12)*, pp. 119–126, IEEE, Wuxi, China, 2012.

[22] M. Kovatsch, S. Mayer, and B. Ostermaier, "Moving application logic from the firmware to the cloud: towards the thin server architecture for the internet of things," in *Proceedings of the 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS '12)*, pp. 751–756, July 2012.

[23] T. Riedel, N. Fantana, A. Genaid, D. Yordanov, H. R. Schmidtke, and M. Beigl, "Using web service gateways and code generation for sustainable IoT system development," in *Proceedings of the Internet of Things (IoT '10)*, pp. 1–8, IEEE, Tokyo, Japan, December 2010.

[24] F. Tao, Y. Cheng, L. D. Xu, L. Zhang, and B. H. Li, "CCIoT-CMfg: cloud computing and internet of things-based cloud manufacturing service system," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1435–1442, 2014.

[25] L. Gao, T. H. Luan, B. Liu, W. Zhou, and S. Yu, "Fog computing and its applications in 5g," in *5G Mobile Communications*, pp. 571–593, Springer, 2017.

[26] C. Dsouza, G.-J. Ahn, and M. Taguinod, "Policy-driven security management for fog computing: preliminary framework and a case study," in *Proceedings of the 15th IEEE International Conference on Information Reuse and Integration (IEEE IRI '14)*, pp. 16–23, August 2014.

[27] L. Prieto González, C. Jaedicke, J. Schubert, and V. Stantchev, "Fog computing architectures for healthcare: wireless performance and semantic opportunities," *Journal of Information, Communication and Ethics in Society*, vol. 14, no. 4, pp. 334–349, 2016.

[28] R. Chen, "An intelligent value stream-based approach to collaboration of food traceability cyber physical system by fog computing," *Food Control*, vol. 71, pp. 124–136, 2017.

[29] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. We, and L. Sun, "A view of fog computing from networking perspective," https://arxiv.org/abs/1602.01509.

[30] R. Zgheib, E. Conchon, and R. Bastide, "Engineering IoT healthcare applications: towards a semantic data driven sustainable architecture," in *eHealth 360◦*, vol. 181 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 407–418, Springer International Publishing, Cham, 2017.

[31] P. K. Gupta, B. T. Maharaj, and R. Malekian, "A novel and secure IoT based cloud centric architecture to perform predictive analysis of users activities in sustainable health centres," *Multimedia Tools and Applications*, 2016.

[32] R. Barbosa, D. Nunes, A. Figueira et al., "An architecture for emotional smartphones in Internet of Things," in *Proceedings of the IEEE Ecuador Technical Chapters Meeting (ETCM '16)*, vol. 1, pp. 1–5, Guayaquil, Ecuador, October 2016.

[33] C. Sarkar, A. U. N. Sn, R. V. Prasad, A. Rahim, R. Neisse, and G. Baldini, "Diat: a scalable distributed architecture for iot," *IEEE Internet of Things Journal*, vol. 2, no. 3, pp. 230–239, 2015.

[34] O. Flauzac, C. Gonzalez, A. Hachani, and F. Nolot, "SDN based architecture for IoT and improvement of the security," in *Proceedings of the 29th IEEE International Conference on Advanced Information Networking and Applications Workshops (WAINA '15)*, pp. 688–693, March 2015.

[35] F. Y. Okay and S. Ozdemir, "A fog computing based smart grid model," in *Proceedings of the International Symposium on Networks, Computers and Communications (ISNCC '16)*, pp. 1–6, Yasmine Hammamet, Tunisia, May 2016.

[36] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, and Q. Yang, "A hierarchical distributed fog computing architecture for big data analysis in smart cities," in *Proceedings of the ASE BigData and SocialInformatics (ASE BD and SI '15)*, ACM, October 2015.

[37] M. A. Al Faruque and K. Vatanparvar, "Energy management-as-a-service over fog computing platform," *IEEE Internet of Things Journal*, vol. 3, no. 2, pp. 161–169, 2016.

[38] J. Zhu, D. S. Chan, M. S. Prabhu, P. Natarajan, H. Hu, and F. Bonomi, "Improving web sites performance using edge servers in fog computing architecture," in *Proceedings of the IEEE 7th International Symposium on Service-Oriented System Engineering (SOSE '13)*, pp. 320–323, IEEE, San Francisco, Calif, USA, March 2013.

[39] M. Aazam and E.-N. Huh, "Fog computing and smart gateway based communication for cloud of things," in *Proceedings of the 2nd International Conference on Future Internet of Things and Cloud (FiCloud '14)*, pp. 464–470, Barcelona, Spain, August 2014.

[40] C.-X. Wang, F. Haider, X. Gao et al., "Cellular architecture and key technologies for 5G wireless communication networks," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 122–130, 2014.

[41] Y. Hu, H. Li, Z. Chang, and Z. Han, "Scheduling strategy for multimedia heterogeneous high-speed train networks," *IEEE Transactions on Vehicular Technology*, 2016.

[42] L. Lei, J. Lu, Y. Jiang et al., "Stochastic delay analysis for train control services in next-generation high-speed railway communications system," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 1, pp. 48–64, 2016.

[43] B. Hui, J. Kim, H. Chung, and I. Kim, "Creation and control of handover zone using antenna radiation pattern for high-speed train communications in unidirectional networks," in *Proceedings of the International Conference on Information and Communication Technology Convergence (ICTC '16)*, pp. 737–740, IEEE, Jeju, Korea, October 2016.

[44] Z. Li, Y. Chen, H. Shi, and K. Liu, "NDN-GSM-R: a novel high-speed railway communication system via named data networking," *EURASIP Journal on Wireless Communications and Networking*, vol. 2016, no. 1, article 48, pp. 1–5, 2016.

[45] E. A. Ibrahim, E. F. Badran, and M. R. Rizk, "An optimized LTE measurement handover procedure for high speed trains using WINNER II channel model," in *Proceedings of the 22nd Asia-Pacific Conference on Communications (APCC '16)*, pp. 197–203, IEEE, Yogyakarta, Indonesia, August 2016.

[46] S. Xu, G. Zhu, B. Ai, and Z. Zhong, "A survey on high-speed railway communications: a radio resource management perspective," *Computer Communications*, vol. 86, pp. 12–28, 2016.

[47] L. Gao, T. H. Luan, S. Yu, W. Zhou, and B. Liu, "FogRoute: DTN-based data dissemination model in fog computing," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 225–235, 2016.

[48] H. Kopetz and S. Poledna, "In-vehicle real-time fog computing," in *Proceedings of the 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W '16)*, pp. 162–167, Toulouse, France, June 2016.

[49] O. Goldschmidt, P. Jaillet, and R. LaSota, "On reliability of graphs with node failures," *Networks*, vol. 24, no. 4, pp. 251–259, 1994.

[50] A. Wald, "Statistical decision functions," *Annals of Mathematical Statistics*, vol. 20, pp. 165–205, 1949.

[51] V. Sharma and R. Kumar, "A cooperative network framework for multi-UAV guided ground ad hoc networks," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 77, no. 3-4, pp. 629–652, 2015.

[52] V. Sharma, M. Bennis, and R. Kumar, "UAV-assisted heterogeneous networks for capacity enhancement," *IEEE Communications Letters*, vol. 20, no. 6, pp. 1207–1210, 2016.

[53] K. Zhang, X. Liang, R. Lu, and X. Shen, "Sybil attacks and their defenses in the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 5, pp. 372–383, 2014.

[54] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDOS) flooding attacks," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013.

[55] T. Booth and K. Andersson, "Network security of internet services: eliminate DDoS reflection amplification attacks," *Journal of Internet Services and Information Security*, vol. 5, pp. 58–79, 2015.

[56] L. Wallgren, S. Raza, and T. Voigt, "Routing attacks and countermeasures in the RPL-based internet of things," *International Journal of Distributed Sensor Networks*, vol. 2013, Article ID 794326, 2013.

Hindawi
Mobile Information Systems
Volume 2017, Article ID 9816467, 8 pages
https://doi.org/10.1155/2017/9816467

Research Article

Joint Optimized CPU and Networking Control Scheme for Improved Energy Efficiency in Video Streaming on Mobile Devices

Sung-Woong Jo and Jong-Moon Chung

School of Electrical and Electronic Engineering, Yonsei University, 50 Yonsei-ro, Seodaemun-gu, Seoul 03722, Republic of Korea

Correspondence should be addressed to Jong-Moon Chung; jmc@yonsei.ac.kr

Received 30 December 2016; Accepted 26 February 2017; Published 16 March 2017

Academic Editor: Karl Andersson

Video streaming service is one of the most popular applications for mobile users. However, mobile video streaming services consume a lot of energy, resulting in a reduced battery life. This is a critical problem that results in a degraded user's quality of experience (QoE). Therefore, in this paper, a joint optimization scheme that controls both the central processing unit (CPU) and wireless networking of the video streaming process for improved energy efficiency on mobile devices is proposed. For this purpose, the energy consumption of the network interface and CPU is analyzed, and based on the energy consumption profile a joint optimization problem is formulated to maximize the energy efficiency of the mobile device. The proposed algorithm adaptively adjusts the number of chunks to be downloaded and decoded in each packet. Simulation results show that the proposed algorithm can effectively improve the energy efficiency when compared with the existing algorithms.

1. Introduction

Due to rapid advances in wireless networks and data processing on mobile devices, a continuously growing number of users are accessing high quality video streaming services from mobile devices [1, 2]. According to [2], video streaming services over wireless networks will generate three-quarters of the mobile data traffic and increase ten times between 2014 and 2019. However, such video streaming services consume a significant amount of energy, resulting in a reduced battery lifetime. This is a critical problem especially for mobile devices, due to their limited energy supply. Therefore, minimizing the energy consumption of mobile video streaming services has become an essential research field.

In mobile video streaming services, the network interface and central processing unit (CPU) are two major energy consuming components, which support data communication and video data decoding [3]. Therefore, there have been many studies which focus on minimizing the energy consumption of the network interface and/or CPU. For the network interface, buffer management schemes were proposed to reduce the energy consumption of mobile devices [4, 5]. By dynamically controlling downloading activities, these schemes improved the energy efficiency of video streaming services on mobile devices. In addition, in order to deal with both cellular and Wi-Fi networks, energy efficient methods that control both LTE and Wi-Fi simultaneously in supporting video streaming services over heterogeneous wireless networks were proposed [6, 7]. Meanwhile, for the CPU, dynamic voltage and frequency scaling (DVFS), which can save energy at the cost of a reduced processing speed, is widely used on mobile devices. Based on using the DVFS technique, to improve energy saving of video applications, power management methods were proposed in [8, 9]. These algorithms adaptively control the processing speed through predicting the decoding time of the video. In addition, several research works have focused on scheduling and DVFS in multicore processors [10–12]. Furthermore, for energy optimization among different components including the network interface and CPU, a cross-layer framework was proposed in [13]. The scheme of [13] can reduce the energy consumption of the network interface and CPU. However,

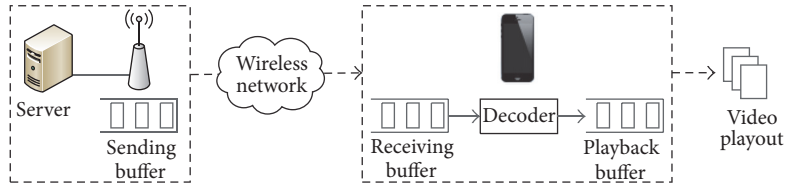


Figure 1: System model of video streaming services on mobile devices.

none of these papers deal with joint optimization of the network interface and CPU. Additional energy saving can be achievable from approaches that jointly consider the trade-off between the energy consumption and the processing time both on the network interface and on the CPU.

In this paper, a joint optimization scheme for improved energy efficiency supporting mobile video streaming services is proposed. First, the energy consumption of the network interface and CPU is analyzed while receiving and decoding the video chunks, respectively. Then, a joint optimization problem statement maximizing the total energy efficiency of the network interface and CPU is formulated. The proposed scheme solves the joint optimization problem and uses the solution to adaptively control the number of chunks to be downloaded and decoded in each packet by considering the video characteristics, wireless network status, and buffer status. The energy consumption of the proposed algorithm is compared with four existing algorithms based on various environments. Simulation results show that the proposed Joint optimized CPU and Video streaming Network (JCVN) control algorithm can improve the energy efficiency of video streaming services on mobile devices.

2. System Model

2.1. Mobile Video Streaming Service Model. In Figure 1, the system model is illustrated [5], where a mobile device directly obtains video streams from the streaming server through the wireless network interface. The video streaming service is assumed to be a video sequence that is compressed at a constant bit rate r bits/s. The video sequence is divided into equal-size chunks of rT_0 bits sent periodically in packets with payload durations of T_0 s. The system is assumed to operate in discrete time units based on packets $i = \{1, 2, \ldots, T\}$, where T is the maximum number of packets needed to transmit the entire video sequence. By considering a time-varying wireless channel, the download data rate of the mobile device at packet i is denoted as $R_d(i)$. It is assumed that the download data rate remains constant within a packet duration but can vary across different packets.

A mobile user may experience interruptions during video playback when the download rate of the video stream is lower than the encoding rate of the video stream due to the time-varying wireless channel and network conditions. Therefore, mobile devices typically use a receiving buffer and playback buffer to satisfy the QoS requirement (i.e., no interruption) [14]. In a mobile device, the downloaded data chunks enter the receiving buffer and the chunks are decoded in the video decoder. The decoded chunks enter the playback buffer and they are displayed corresponding to the playback rate, which is equivalent to the video encoding rate.

Let $B_r(i)$ and $B_p(i)$ denote the number of chunks in the receiving buffer and playback buffer at the beginning of packet i, respectively. Because mobile devices have limited resources, $B_r(i)$ and $B_p(i)$ are bounded by their maximum thresholds as $0 \le B_r(i) \le B_{r,\mathrm{Th}}$ and $0 \le B_p(i) \le B_{p,\mathrm{Th}}$, respectively. In addition, the processing delay in the video decoder should be taken into consideration when the chunks are decoded. The processing delay is dependent on the CPU frequency at which the video decoder decodes the chunks. The processing delay is inversely proportional to the CPU frequency [8, 9, 13]. Let D_p denote the processing delay required to decode a chunk at the highest CPU frequency $f_{c,\max}$. Then, the processing delay required to decode a chunk at CPU frequency $f_c(i)$ in packet i is $D_p f_{c,\max}/f_c(i)$.

2.2. Energy Consumption Model. The energy consumption of the wireless network interface card (WNIC) is modeled as $E = P_n t + P_d l$, where P_n is the average power consumption in active state, t is the time duration in active state, P_d is the average power consumption per downloaded data, and l is the amount of downloaded data [5]. By using the WNIC's energy consumption model, in order to evaluate the energy efficiency, the energy consumption gain of the WNIC for packet i is defined as

$$\begin{aligned} E_n(i) &= \left(P_n n_d(i)\frac{rT_0}{R_{\min}} + P_d n_d(i) rT_0\right) \\ &\quad - \left(P_n n_d(i)\frac{rT_0}{R_d(i)} + P_d n_d(i) rT_0\right) \\ &= P_n n_d(i)\left(\frac{rT_0}{R_{\min}} - \frac{rT_0}{R_d(i)}\right), \end{aligned} \tag{1}$$

where $n_d(i)$ is the number of chunks downloaded in packet i. The energy consumption gain of the WNIC $E_n(i)$ is the amount of energy saving that can be achieved when downloading $n_d(i)$ chunks at the download rate of $R_d(i)$ compared to the case when downloading equal-size chunks at the minimum download rate of $R_{\min}$. Note that, in (1), the energy consumption gain of the WNIC $E_n(i)$ cannot be a negative value due to $R_{\min} \le R_d(i)$.

Furthermore, the power consumption of the CPU is a function of the CPU frequency. According to the model in [15], the average power consumption of the CPU at CPU frequency $f_c(i)$ in packet i can be defined as $P_p(i) = \alpha f_c(i)^3 + \beta$, where α and β are constants that depend on the mobile

device. Then, similar to the WNIC's energy consumption gain of (1), the energy consumption gain of the CPU based on packet i is defined as

$$E_p(i) = n_p(i) D_p \left(\left(\alpha f_{c,\max}^3 + \beta \right) - \left(\alpha f_c(i)^3 + \beta \right) \frac{f_{c,\max}}{f_c(i)} \right), \quad (2)$$

where $n_p(i)$ is the number of chunks decoded in packet i. The energy consumption gain of the CPU $E_p(i)$ is the amount of energy saving that can be achieved when decoding $n_p(i)$ chunks at the CPU frequency of $f_c(i)$ compared to the case when decoding equal-size chunks at the maximum CPU frequency of $f_{c,\max}$.

3. JCVN Controller Design

The objective of this paper is to maximize the total energy consumption gain of the WNIC and CPU while satisfying the QoS requirements of the video streaming service. Based on the mathematical models of the previous section, the optimization problem can be stated as

$$\max_{n_d(i), n_p(i)} \quad E_n(i) + E_p(i), \quad \forall i \quad (3)$$

$$\text{s.t.} \quad 0 \le n_d(i) \le \frac{R_d(i)}{r}, \quad (4)$$

$$0 \le n_p(i) \le \frac{T_0}{D_p}, \quad (5)$$

$$0 \le B_r(i+1) \le B_{r,\text{Th}}, \quad (6)$$

$$0 \le B_p(i+1) \le B_{p,\text{Th}}, \quad (7)$$

where the number of chunks in the receiving buffer and playback buffer can be updated as follows:

$$B_r(i+1) = B_r(i) + n_d(i) - n_p(i), \quad B_p(i+1) = B_p(i) + n_p(i) - 1. \quad (8)$$

In (8), $n_d(i)$ is the number of chunks arriving at the receiving buffer through downloading of packet i, $n_p(i)$ is the number of chunks migrating from the receiving buffer to the playback buffer through the decoding process of packet i, and $rT_0/rT_0 = 1$ is the number of chunks to be displayed to satisfy the QoS requirements based on the encoding rate, in which the numerator term rT_0 is the amount of video data to be displayed at the encoding rate r in a packet duration T_0 and the denominator term rT_0 is the size of a video chunk.

Constraint (4) ensures that the number of chunks to be downloaded is not a negative value and cannot be greater than the maximum number of chunks which the mobile device can download at its download rate in a packet. Constraint (5) ensures that the number of chunks to be decoded is not a negative value and cannot be greater than the maximum number of chunks which can be decoded at the maximum CPU frequency in a packet. Constraints (6) and (7) ensure that the numbers of chunks in the receiving buffer and playback buffer are not negative values and should also be less than their maximum thresholds. That is, JCVN controls the number of chunks to be downloaded and decoded while avoiding buffer overflow. In addition, constraint (7) ensures that the video chunks are displayed at their encoding rate and the viewer will not experience any interruption in the playback. This is because $B_p(i+1) \ge 0$ (i.e., $B_p(i) + n_p(i) \ge 1$) guarantees that the playback buffer will maintain more chunks than the number of chunks to be displayed at the encoding rate.

In addition, to maximize the energy consumption gain of the CPU, when the CPU decodes $n_p(i)$ video chunks, CPU frequency $f_c(i)$ in packet i should be selected to the minimum CPU frequency at which the CPU can decode $n_p(i)$ chunks in packet i. Therefore, the number of chunks to be decoded can be presented as a function of the CPU frequency in packet i as follows:

$$n_p(i) = \begin{cases} \left(T_0 - \dfrac{rT_0}{R_d(i)} \right) \dfrac{f_c(i)}{D_p f_{c,\max}} & B_r(i) = 0 \\ \dfrac{T_0 f_c(i)}{D_p f_{c,\max}} & B_r(i) > 0, \end{cases} \quad (9)$$

which indicates that if there is no chunk in the receiving buffer (i.e., $B_r(i) = 0$), the decoding process starts after downloading a chunk. Otherwise, if there are chunks in the receiving buffer (i.e., $B_r(i) > 0$), the chunks are decoded during the packet length T_0. In addition, (9) satisfies constraint (5).

By solving the optimization problem (3)–(7), an optimal solution set of $n_d(i)$ and $n_p(i)$ which maximizes the total energy consumption gain of the WNIC and CPU based on each packet i can be obtained. The optimization problem is transformed into two variables x and y as follows:

$$\begin{aligned} \max_{x,y} \quad & f(x,y) = ax + \left(-by^3 + cy - d \right) \\ \text{s.t.} \quad & 0 \le x \le e, \\ & 0 \le y \le g, \\ & h \le x - y \le k, \\ & l \le y \le m, \end{aligned} \quad (10)$$

where $x = n_d(i)$, $y = n_p(i)$, and a, b, c, d, e, g, h, k, l, and m are (for the ith packet) constant values as

$$a = P_n T_0 r \left(\frac{1}{R_{\min}} - \frac{1}{R_d(i)} \right),$$

$$b = \frac{\alpha T_0 \left(D_p f_{c,\max}/T_0 \right)^3}{\left(1 - r/R_d(i) \right)^2} \quad \left(B_r(i) = 0 \right),$$

$$b = \alpha T_0 \left(\frac{D_p f_{c,\max}}{T_0} \right)^3 \quad \left(B_r(i) > 0 \right),$$

(1) GET *input parameters of the energy consumption model*
(2) COMPUTE *optimal candidate solution set* $(n_{d,1}(i)^*, n_{p,1}(i)^*)$ *for Case 1*
(3) COMPUTE *optimal candidate solution set* $(n_{d,2}(i)^*, n_{p,2}(i)^*)$ *for Case 2*
(4) COMPARE *values of the objective function at two candidate solution sets*
(5) if $E_n(n_{d,1}(i)^*) + E_p(n_{p,1}(i)^*) \geq E_n(n_{d,2}(i)^*) + E_p(n_{p,2}(i)^*)$
(6) SET *optimal solution set* to $(n_{d,1}(i)^*, n_{p,1}(i)^*)$
(7) else
(8) SET *optimal solution set* to $(n_{d,2}(i)^*, n_{p,2}(i)^*)$
(9) end if

Algorithm 1: JCVN algorithm.

$$
\begin{aligned}
c &= \left(\alpha f_{c,\max}^3 + \beta\right) D_p, \\
d &= T_0 \beta \left(1 - \frac{r}{R_d(i)}\right) \quad \left(B_r(i) = 0\right), \\
d &= T_0 \beta \quad \left(B_r(i) > 0\right), \\
e &= \frac{R_d(i)}{r}, \\
g &= \frac{T_0}{D_p}, \\
h &= -B_r(i), \\
k &= B_{r,\mathrm{Th}} - B_r(i), \\
l &= -B_p(i) + 1, \\
m &= B_{p,\mathrm{Th}} - B_p(i) + 1.
\end{aligned}
\tag{11}
$$

Let set S denote the range of (x, y) satisfying the above constraints. Then, through the *Closed Interval Method* of [16], the maximum value of $f(x, y)$ can be obtained at the critical points in S or at the extreme values on the boundary of S. In addition, because $f(x, y)$ is a linear function with respect to x, $f(x, y)$ is maximized at the maximum value among the range of x with an arbitrarily fixed value of y. Therefore, an optimal solution set (x, y) exists on line of $x = e$ (Case 1) or $y = x - k$ (Case 2).

Case 1. From $x = e$, the objective function is represented as a function of y, which is $f(e, y) = -by^3 + cy + ae - d$ with constraint of $\max[0, e - k, l] \leq y \leq \min[g, e - h, m]$. By letting $\partial f(e, y)/\partial y = 0$, the only critical point of the equation is computed as $\sqrt{c/3b}$. In addition, because *the second derivative test* gives $\partial^2 f(e, y)/\partial y^2 = -6by < 0$ at $\sqrt{c/3b}$, the objective function is concave. Therefore, the objective function has a maximum value at the critical point of $\sqrt{c/3b}$, or the minimum or maximum endpoint of y [16]. If the critical point exists on the larger value than the range of y (i.e., $\min[g, e-h, m] < \sqrt{c/3b}$), the objective function has a maximum value at the maximum end point of $y = \min[g, e-h, m]$. Otherwise, if the critical point exists on the smaller value than the range of y (i.e., $\max[0, e - k, l] > \sqrt{c/3b}$), the objective function has a maximum value at the minimum end point of $y = \max[0, e - k, l]$. Otherwise, if the critical point exists in the range of y (i.e., $\max[0, e - k, l] \leq \sqrt{c/3b} \leq \min[g, e-h, m]$), the objective function has a maximum value at the critical point of $\sqrt{c/3b}$.

Case 2. From $x = y + k$, the objective function is also represented as a function of y, which is $f(y + k, y) = -by^3 + (a + c)y + ak - d$ with a constraint of $\max[0, l] \leq y \leq \min[e - k, g, m]$. In a similar way, to solve Case 2, by letting $\partial f(y + k, y)/\partial y = 0$, the only critical point of the equation is computed as $\sqrt{(a + c)/3b}$. Then, the objective function has a maximum value at the maximum end point of $y = \min[e - k, g, m]$, or the minimum end point of $y = \max[0, l]$, or the critical point of $\sqrt{(a + c)/3b}$.

Therefore, an optimal solution set of $n_d(i)^*$ and $n_p(i)^*$ which maximizes the objective function can be obtained by comparing the values of $f(x, y)$ at the two candidate solutions from Cases 1 and 2. Note that the optimal solution set of $n_d(i)^*$ and $n_p(i)^*$ is obtained by assuming that the objective function is a continuous function of $n_d(i)$ and $n_p(i)$. Therefore, because the numbers of chunks to be downloaded and decoded are positive integer values, in the performance evaluation, the values of the optimal solution set were selected to be positive integer values that satisfy the constraints of the optimization problem and have a minimum difference with the derived optimal solution set of $n_d(i)^*$ and $n_p(i)^*$, respectively.

The pseudocode of the JCVN optimization algorithm is presented in Algorithm 1. First, the input parameters of the energy consumption model are obtained based on the optimization problem (step (1)). JCVN computes the optimal candidate solution set of the number of chunks to be downloaded and decoded for Cases 1 and 2 (steps (2) and (3)). Then, an optimal solution set is determined by comparing the values of the objective function at the two candidate solution sets from Cases 1 and 2. If the value of the objective function at Case 1's optimal candidate solution is equal to or larger than the value of the objective function at Case 2's optimal candidate solution (i.e., $E_n(n_{d,1}(i)^*) + E_p(n_{p,1}(i)^*) \geq E_n(n_{d,2}(i)^*) + E_p(n_{p,2}(i)^*)$), the optimal solution is selected as the candidate solution set for Case 1 (steps (5) and (6)). Otherwise, if the value of the objective function at Case

Table 1: Simulation parameter settings.

Parameters	Values
Download rate of a mobile device	A Rayleigh distributed wireless channel with the means of 0.5, 1, 1.5, and 2 Mbps
Video encoding rate	320 kbps
Entire video length	200 s
Packet length	2 s
Maximum receiving and playback buffer size	20 chunks
Average power consumption in active state (P_n)	0.897 W
Average power consumption per downloaded data (P_d)	0.134 μJ
CPU frequency	0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, and 1.6 GHz
Coefficient for CPU's power consumption (α)	0.4 s
Coefficient for CPU's power consumption (β)	0.3 s
Processing density	1000 cycles/bit

1's optimal candidate solution is less than the value of the objective function at Case 2's optimal candidate solution (i.e., $E_n(n_{d,1}(i)^*) + E_p(n_{p,1}(i)^*) < E_n(n_{d,2}(i)^*) + E_p(n_{p,2}(i)^*)$), the optimal solution is selected as the candidate solution set for Case 2 (steps (7) and (8)).

4. Performance Evaluation

4.1. Simulation Settings. In order to make an accurate performance comparison, the experiment parameters of [5, 15] were applied in the simulation experiments as described in the following. The video encoding rate was set to 320 kbps and the video length was 200 s. The entire video file was divided into packets of 2 s, where the mobile device's receiving buffer and playback buffer can contain up to 20 chunks of video data. In the simulation, a Rayleigh distributed wireless channel was applied to represent the multipath fading characteristics, which results in variations of the download rate of video data with the means of 0.5, 1, 1.5, and 2 Mbps [5]. For example, the variation of the download rates with a mean of 1.5 Mbps is presented in Figure 2(a). The WNIC's power consumption parameter values applied in the simulation are $P_n = 0.897$ W and $P_d = 0.134\,\mu$J/bit. In addition, the average number of CPU cycles required per bit (when the application is processed by the CPU) is defined as the processing density, which is dependent on the application type. For the video streaming simulations, a mixture of H.264 240p, 480p, and 720p videos at 30 fps based on a closed group of pictures (GoP) without coordinated universal time (UTC) was used. In addition, a 1,000-cycle/bit processing density was applied based on [15], and the CPU's power consumption model uses the parameter values of $\alpha = 0.4$ and $\beta = 0.3$. The detailed simulation parameter settings are summarized in Table 1.

In the performance analysis, two representative WNIC downloading schemes and two of the most popular CPU processing power control schemes that are used in smart devices are compared with the proposed JCVN scheme. The two representative WNIC downloading schemes are Aggressive Buffer Management (ABM) and Periodical Buffer Management (PBM) [5]. In ABM, the video application actively downloads video chunks until the receiving buffer of the mobile device is full, and is popularly used in mobile devices with unlimited (or very large) power supplies and has no constraints applied to the video streaming services. In PBM, the video application makes requests for a fixed number of video chunks to be periodically sent from the server and stops downloading if the requested video chunks have been all received within the time limit or if the mobile device's receiving buffer is full, which is a very popular WNIC scheme applied in mobile devices. In the simulation experiments, the PBM default period T_0 is set based on the viewer requests of two, three, four, and five video chunks per period for the mean download rates of 0.5, 1, 1.5, and 2 Mbps, respectively.

The two most popular CPU processing power control schemes are the maximum CPU frequency (MAX) scheme and the dynamic voltage and frequency scaling (DVFS) scheme. In MAX, the CPU processes video chunks at its highest frequency if video chunks to be decoded are in its receiving buffer. This strategy provides a high performance by processing data at the highest CPU frequency, while it consumes a lot of energy. In DVFS, the CPU frequency is adjusted according to the amount of video chunks to be decoded in the receiving buffer. When DVFS is used, the CPU frequency is set to maximum when the number of video chunks to be decoded is greater than a threshold, and the CPU frequency proportionally decreases (in a staircase fashion) as the decoding load diminishes. The threshold of DVFS was set to four video chunks in the experiments. Note that when the receiving buffer and playback buffer become full, the WNIC and CPU of all existing schemes and JCVN are switched to the power-saving mode in which they do not perform downloading and decoding.

In the simulations, the two WNIC schemes were each teamed up with the two CPU control schemes in the following form of PBM with MAX, ABM with MAX, PBM with DVFS, and ABM with DVFS, which are, respectively, denoted as PBM + MAX, ABM + MAX, PBM + DVFS, and ABM + DVFS in Figures 2 and 3.

4.2. Performance Analysis. Figure 2 illustrates the difference of JCVN (compared to other schemes) in how it jointly controls the number of chunks to be downloaded and

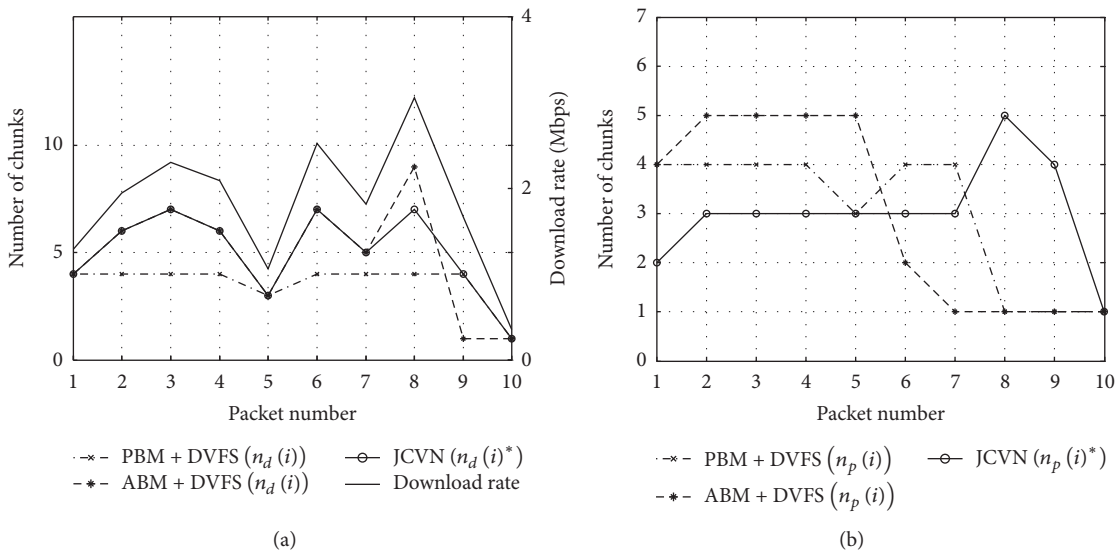


Figure 2: (a) Number of chunks downloaded per packet. (b) Number of chunks decoded per packet.

decoded to maximize the combined energy efficiency of the WNIC and CPU. Figures 2(a) and 2(b) show the number of chunks to be downloaded and decoded for each packet at the mean download rate of 1.5 Mbps. In Figure 2(a), ABM + DVFS and JCVN download chunks in accordance with the download rate, while PBM + DVFS has little variation in the number of downloaded chunks regardless of the download rate. Therefore, ABM + DVFS and JCVN can provide a gain in reducing the WNIC's energy consumption compared to PBM + DVFS. In Figure 2(b), it is observed that PBM + DVFS and ABM + DVFS aggressively decode chunks at the beginning of a download session (i.e., packets 1~5) quickly filling up their buffers and also use full CPU processing capability in the decoding process. Compared to this, JCVN attempts to save energy by downloading only at a sufficient rate in the initial stages (i.e., packets 1~5) and also selects a sufficient CPU operational frequency that will allow an above-QoS level of video playout.

In Figure 3, the proposed JCVN scheme is compared with the four schemes of PBM + MAX, ABM + MAX, PBM + DVFS, and ABM + DVFS for various download rates, where Figures 3(a), 3(b), and 3(c) present the mobile device's energy consumption of the WNIC, CPU, and the combined value of the WNIC and CPU, respectively. Figure 3(a) shows that, for all schemes, the energy consumption of the WNIC decreases to around 61% as the average download rate increases. This is because the energy consumption of the WNIC is inversely linear to the download rate as denoted in (1). In Figure 3(b), it is noted that PBM + DVFS and ABM + DVFS consume less energy at the CPU than PBM + MAX and ABM + MAX, which is due to the energy saving capability of DVFS, where the energy saving gain increases as the average download rate decreases. This is because the mobile device can download more video chunks at the receiving buffer in a packet when the download rate is high. Therefore, DVFS will use higher CPU frequencies in order to process the video chunks of the receiving buffer, thereby consuming more energy. Figure 3(c) points out that the sum of the energy consumption of the WNIC and CPU for the proposed JCVN scheme is the lowest among all the algorithms under different download rates. For example, when compared with PBM + MAX and PBM + DVFS, the JCVN scheme can reduce the energy consumption by 10.59~20.05% and 4.08~11.61%, respectively. Because the JCVN scheme adaptively adjusts the number of video chunks to be downloaded and decoded according to the wireless network status as well as the receiving and playback buffer status, it can effectively reduce the total energy consumption of the WNIC and CPU.

5. Conclusion

In order to deal with the energy consumption issue on mobile devices supporting video streaming services, in this paper, a joint optimization algorithm that can reduce the combined energy consumption of the WNIC and CPU is proposed. The energy consumption of the WNIC and CPU is analyzed and the proposed algorithm adaptively adjusts the number of video chunks to be downloaded and decoded in each packet by solving an optimization problem according to the wireless network status as well as the receiving and playback buffer status. The energy consumption of the proposed algorithm is compared with four existing algorithms based on various average download rates. Results show that the proposed JCVN algorithm is more energy efficient compared to other control schemes.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

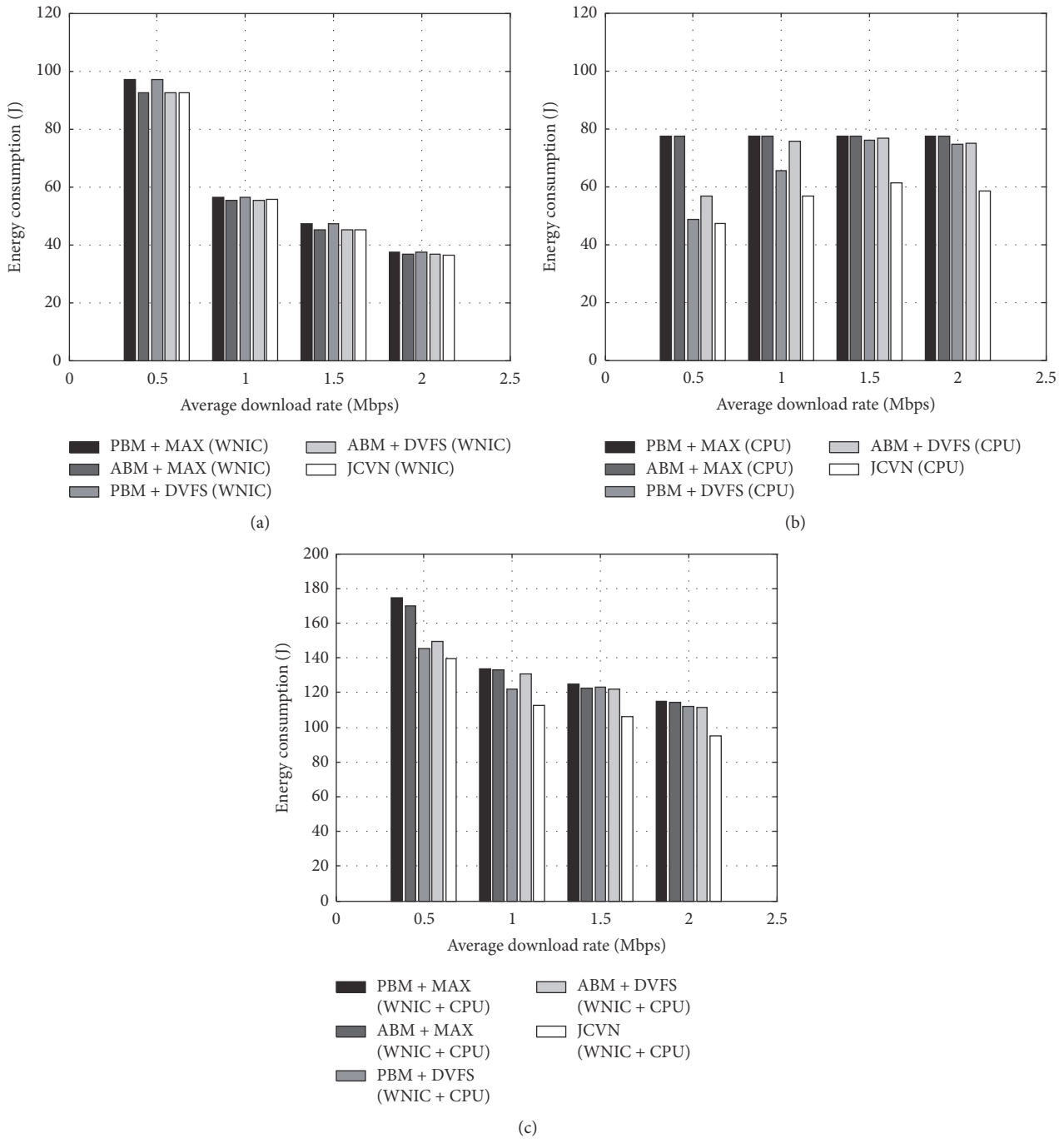


Figure 3: Energy consumption of video streaming service of the (a) WNIC, (b) CPU, and (c) WNIC and CPU combined.

Acknowledgments

This work was supported by a grant from the Disaster and Safety Management Institute funded by the Ministry of Public Safety and Security of the Korean Government (MPSS-SD-2015-41).

References

[1] M. N. Ismail, R. Ibrahim, and M. F. Md. Fudzee, "A survey on content adaptation systems towards energy consumption awareness," *Advances in Multimedia*, vol. 2013, Article ID 871516, 8 pages, 2013.

[2] CISCO, "Cisco visual networking index: global mobile data traffic forecast update, 2014–2019," White Paper, 2015.

[3] M. Kennedy, A. Ksentini, Y. Hadjadj-Aoul, and G.-M. Muntean, "Adaptive energy optimization in multimedia-centric wireless devices: a survey," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 2, pp. 768–786, 2013.

[4] X. Li, M. Dong, Z. Ma, and F. C. A. Fernandes, "GreenTube: power optimization for mobile videostreaming via dynamic

cache management," in *Proceedings of the 20th ACM International Conference on Multimedia (MM '12)*, pp. 279–288, ACM, Nara, Japan, November 2012.

[5] J. He, Z. Xue, D. Wu, D. O. Wu, and Y. Wen, "CBM: online strategies on cost-aware buffer management for mobile video streaming," *IEEE Transactions on Multimedia*, vol. 16, no. 1, pp. 242–252, 2014.

[6] S. Moon, J. Yoo, and S. Kim, "Optimal multi-interface selection for mobile video streaming in efficient battery consumption and data usage," *Mobile Information Systems*, vol. 2016, Article ID 4871629, 18 pages, 2016.

[7] D. H. Bui, K. Lee, S. Oh et al., "GreenBag: energy-efficient bandwidth aggregation for real-time streaming in heterogeneous mobile wireless networks," in *Proceedings of the IEEE 34th Real-Time Systems Symposium (RTSS '13)*, pp. 57–67, December 2013.

[8] X. Liu, P. Shenoy, and M. D. Corner, "Chameleon: application-level power management," *IEEE Transactions on Mobile Computing*, vol. 7, no. 8, pp. 995–1010, 2008.

[9] J. Cho, I. Cho, D.-Y. Kim, and B. Lee, "A combined approach for QoS-guaranteed and low-power video decoding," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 2, pp. 651–657, 2011.

[10] J. Luo and N. K. Jha, "Power-efficient scheduling for heterogeneous distributed real-time embedded systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 6, pp. 1161–1170, 2007.

[11] H. F. Sheikh, H. Tan, I. Ahmad, S. Ranka, and P. Bv, "Energy- and performance-aware scheduling of tasks on parallel and distributed systems," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 8, no. 4, article no. 32, 2012.

[12] H. Jeon, W. H. Lee, and S. W. Chung, "Load unbalancing strategy for multicore embedded processors," *IEEE Transactions on Computers*, vol. 59, no. 10, pp. 1434–1440, 2010.

[13] S. Mohapatra, N. Dutt, A. Nicolau, and N. Venkatasubramanian, "DYNAMO: a cross-layer framework for end-to-end QoS and energy optimization in mobile handheld devices," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 4, pp. 722–737, 2007.

[14] M. Li, Z. Guo, R. Y. Yao, and W. Zhu, "A novel penalty controllable dynamic voltage scaling scheme for mobile multimedia applications," *IEEE Transactions on Mobile Computing*, vol. 5, no. 12, pp. 1719–1733, 2006.

[15] J. Kwak, O. Choi, S. Chong, and P. Mohapatra, "Processor-network speed scaling for energy-delay tradeoff in smartphone applications," *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1647–1660, 2016.

[16] J. Stewart, *Metric International Version Calculus*, Early Transcendentals, Thomson Brooks/Cols, 6th edition, 2008.

Hindawi
Mobile Information Systems
Volume 2017, Article ID 8098394, 13 pages
https://doi.org/10.1155/2017/8098394

Research Article

Fuzzy Theory Based Security Service Chaining for Sustainable Mobile-Edge Computing

Guanwen Li, Huachun Zhou, Bohao Feng, Guanglei Li, Taixin Li, Qi Xu, and Wei Quan

School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China

Correspondence should be addressed to Guanwen Li; 16111011@bjtu.edu.cn

Received 9 December 2016; Accepted 16 January 2017; Published 16 February 2017

Academic Editor: Karl Andersson

Mobile-Edge Computing (MEC) is a novel and sustainable network architecture that enables energy conservation with cloud computing and network services offloading at the edge of mobile cellular networks. However, how to efficiently manage various real-time changing security functions is an essential issue which hinders the future MEC development. To address this problem, we propose a fuzzy security service chaining approach for MEC. In particular, a new architecture is designed to decouple the required security functions with the physical resources. Based on this, we present a security proxy to support compatibility to traditional security functions. Furthermore, to find the optimal order of the required security functions, we establish a fuzzy inference system (FIS) based mechanism to achieve multiple optimal objectives. Much work has been done to implement a prototype, which is used to analyze the performance by comparing with a widely used method. The results prove that the proposed FIS mechanism achieves an improved performance in terms of Inverted Generational Distance (IGD) values and execution time with respect to the compared solution.

1. Introduction

With the popularity of massive mobile terminals, the global mobile traffic has an exploding growth in the past years. Cisco statistics show the mobile data traffic grew 74% in 2015 and will further increase nearly eight times from 2016 to 2020 [1]. However, the energy efficiency of traditional mobile network infrastructures cannot afford higher and higher requirements of mobile network services with the increasing of mobile users. To alleviate this problem, Mobile-Edge Computing (MEC) was proposed as a novel and sustainable mobile network framework [2]. Different from the energy-efficient technology in wireless sensor network [3, 4], the main idea of MEC is to deploy Information Technology (IT) based services at the edges of mobile networks, which benefits from the energy management of edge clouds. Based on the technology of cloud computing, it enables mobile computation offloading and improves the energy efficiency of mobile network. Due to the above features, MEC is expected to provide the network environment with low energy consumption and high bandwidth for mobile users.

As for the MEC, energy-efficient computation offloading mechanism is one of the greatest concerns in both academia and industry areas. Orsini et al. [5] presented a programming model for mobile application developers to easily benefit from the computation offloading. Chen et al. [6] proposed a game theoretic algorithm for multiuser computation offloading. Based on a successive convex approximation technique, Sardellitti et al. [7] researched the optimization of radio and computational resources. Beck et al. [8] introduced a way to save power by offloading video transcoding from mobile devices. However, the above researches focus on the common methods of computation offloading. There are few solutions concerned about security issues by using of these methods to save energy and increase mobile traffic.

With the fast development of mobile network, the security requirements are becoming increasingly diverse for mobile traffic. As we known, there is little work focusing on flexible and real-time changing security services to meet various security requirements in MEC. Therefore, our work is motivated to propose a new solution for MEC to satisfy diversity security requirements. We introduce the security service chaining for MEC by referring to the idea of service function chaining (SFC) [9]. The security service chaining architecture can provide dynamically changing security services in terms of mobile user requirements. Moreover, some

network-related vulnerabilities for mobile devices, such as Denial of Service attack [10], can be solved by the security function in our architecture. With the proposed architecture, a complex mobile security service can be split to some existing simple security functions. Due to the high performance and virtualization of the cloud computing, it is flexible to compose several required security functions for different security requirements in MEC.

In general, the main contributions of this paper are fourfold:

(1) We propose the architecture for MEC with security service chaining, which deploys security functions on a mobile-edge cloud for the mobile user equipment. Because the traffic is steered with a standard SFC way, a security function proxy for traditional service functions is also proposed to be compatible with the new architecture. Moreover, the proxy can achieve the conversion from a stateless security function to a stateful one.

(2) A graph theoretic model is used to describe the proposed security service chaining. Because the decision-making process about a security service chain is influenced by many factors, we present a fuzzy inference system (FIS) based algorithm to find the proper order of required security functions.

(3) We implement the security service chain in prototype, including packet routing with Network Service Header (NSH) encapsulation [11] and the proposed security functions proxy. There are lots of work on performance analysis of our FIS based algorithm. To make a comparison, we select a simple additive weighting (SAW) method as for the contrast algorithm. The SAW is a widely used method for multiobjective optimization. The results show that our algorithm achieves a better performance than the SAW in terms of Inverted Generational Distance (IGD) values and execution time.

The remainder of this paper is organized as follows: Section 2 discusses the related work. Section 3 presents the architecture of security service chaining for MEC. In Section 4, we formulate the security service chain by the graph based model and present a fuzzy inference system based algorithm for security service composition. Section 5 introduces the implementation of security service chain in cloud. We also present a comparison algorithm and the corresponding evaluation method to evaluate our proposed algorithm in Section 5. Section 6 concludes this paper.

2. Related Work

Recently, many researches [12–16] have focused on combining different service functions to provide a scalable service chaining for user-defined requirement. The core idea of these works is similar to the SFC. Callegati et al. [12] proposed a solution to achieve a service chaining in a cloud environment. It focused on the design of the controller, and its implementation was tested on the Mininet [13], a popular emulation platform for Software-Defined Network (SDN). Different from [12], Gember et al. [14] proposed a new architecture named Stratos, which dynamically instantiated new middle boxes on demand in SDN. Stratos addressed three main problems in steering traffic to appropriate middleboxes: elastic scaling, middlebox placement and flow distribution. FlowTags architecture proposed in [15] achieved the integrate middleboxes into SDN network. FlowTags reduced the overhead as much as possible compared with traditional SDN mechanism, but needed the FlowTags enhanced middle boxes. Qazi et al. [16] implemented dynamic traffic routing without modifying traditional middleboxes. Taking resource constraints into account, it also proposed an algorithm to generate flow paths and forwarding rules. However, these approaches have little consideration on the services deployment for mobile network.

Security service deployment is important to find a proper security service composition. There have been some Web service composition researches, which tried to use a model of graph theory to describe the QoS aware service composition. Yu and Yuan [17] described the service composition as multiconstraint optimal path problem. Sun et al. [18] proposed an improved shortest path-relax method. Jiang et al. [19] used bipartite graph optimal matching algorithm for service selection. da Silva et al. [20] proposed a graph based particle swarm optimization algorithm. Restricted by the graph theory, many existed researches focus on the single-objective optimization of service composition or convert multiobjectives optimization into a multiconstraints problem. However, the process of making decision about security service composition is actually influenced by many extra factors and some of them are in conflict with each other. The existed single-objective optimization algorithms are not suitable to solve the problem.

The fuzzy theory is widely used to solve multiobjectives optimization, which becomes more and more popular in the past few years. Especially for decision-making process about service composition, there are some typical methods based on fuzzy theory. For example, Kashyap and Tyagi [21] optimized the membership function for an efficient service selection and composition. Bakhshi et al. [22] ranked different composite service based on fuzzification of quality criteria of services. However, the existing researches mainly focus on the expressions of user preferences in fuzzy way, but a few consider the order of required function.

In this paper, in order to provide the diversity security functions for different mobile users flexibly, we present a new architecture combining the idea of service function chaining with MEC. Besides, to find the proper security service composition, we use a graph model to describe the security service deployment. Different from the existed researches, we use fuzzy method with graphic theory to find the proper order of the required security functions. Inspired by the work of Dastjerdi and Buyya [23], we choose a fuzzy inference system to achieve the process of making decision with multiple influential factors.

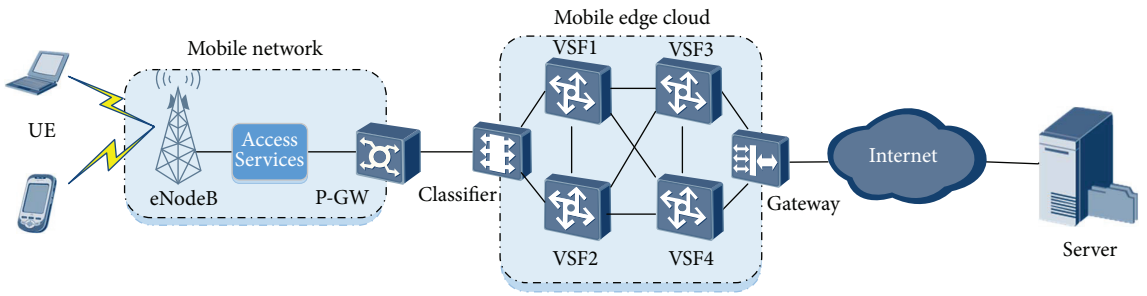


Figure 1: The architecture of mobile-edge security service chaining.

3. Architecture of Security Service Chaining

In this section, we propose an architecture of mobile-edge security service chaining at first. Then, we introduce main elements in the architecture in detail. Particularly, the proposed security function proxy is illustrated by an example for achieving a SFC-aware firewall.

The traditional mobile network is very complex due to integrating many different service functions. For example, in a typical 3GPP-based mobile network, the traffic from the user equipment (UE) is encapsulated in 3GPP specific tunnels terminating eventually at the packet gateway (P-GW). P-GW is the packet gateway of traditional mobile network, which is responsible to forward data to the other network. In other words, all the services are all standardized. However, it is unnecessary for each flow to pass through all security services in certain order, which may affect the resources efficiency in mobile network. Thus, we can classify these services into two categories: the access service and the security service. P-GW is the bridge between both of them. The former is necessary for mobile network. However, the latter can be implemented with the cloud technology to improve resources efficiency. Therefore, in our architecture, we divide the traditional mobile network into two independent networks: the dedicated access network and the mobile-edge cloud.

As shown in Figure 1, the proposed architecture is composed of four main parts: the UE, the simplified mobile network, the mobile-edge cloud, and the servers connected with the Internet. In this solution, the access service from enhanced NodeB (eNodeB) to P-GW is not detailed because the security service is focused on in this paper. An UE is a mobile user. The simplified mobile network is only responsible to user radio access. The mobile-edge network plays an important role in our architecture, where we deploy security functions required by mobile users. There are one classifier, some virtualized security functions (VSFs), and one gateway in mobile-edge network. The server connected with the Internet is the destination for the request of mobile user.

In this case, we assume that a mobile user is trying to use his/her cell phone to access the server pass through the Internet. At first, his/her phone, regarded as the UE, connects to the nearby eNodeB. Then, the data is transmitted from eNodeB to P-GW in the mobile network. In this architecture, the next hop of P-GW should be the mobile-edge network. The data will be assigned a security service chain which depended on the user security requirement. The following sections will describe the main elements of our proposed architecture in detail.

3.1. Access Service. The traditional user access services in mobile network are still necessary. After the eNodeB receives the user traffic from the UE, the mobile network will finish user access and authentication process as usual. Then, the traffic is steered to P-GW. Different from traditional mobile network architecture, P-GW is a communication bridge between the user access services and the mobile-edge cloud. It forwards all traffic from a mobile node to the presented cloud, which can provide the required security service chain on a virtualized platform.

3.2. Security Classifier. The traffic sent from P-GW is received by the classifier at first in the cloud. Once the classifier receives a new traffic flow, it will analyze the characteristic of this flow. The classifier identifies different flows based on their identity/type information (e.g., 5-tuple in TCP/IP). Then, there is a unique flow identifier created for this flow. This identifier is regarded as its service path identifier (SPI), which is used to mark the security service chain for this flow. At the same time, the classifier reports its classification to the control plane. After the control plane resolves the result and assigns a proper security service chain for this flow, the classifier will know its next hop. In terms of the forwarding information from control plane, the classifier encapsulates the flow with corresponding service index (SI), which indicates the first (next hop) virtualized security function. Finally, the flow is forwarded to the next hop by this classifier.

3.3. Virtualized Security Function. The virtualized security function in our architecture is a logical concept, which consists of three physical entities: the service forwarder, the security service proxy, and the security function. The following present each element of the virtualized security function, and the relations among them are illustrated as Figure 2.

3.3.1. Service Forwarder. Forwarding is the only one mission for a service forwarder, and it is a bidirectional process. On one hand, when a service forwarder receives a flow from previous service forwarder or the security classifier,

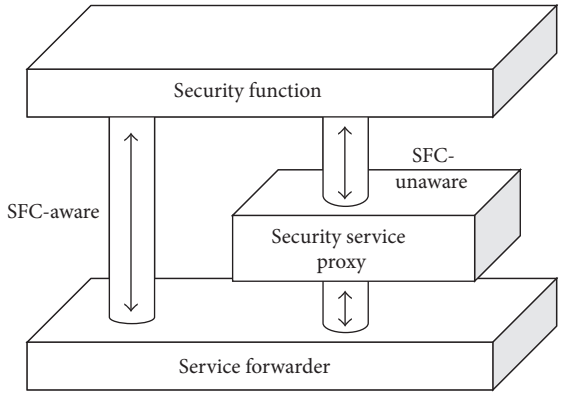


Figure 2: The relations among elements in virtualized security function.

it redirects the flow to the specific security function proxy according to the NSH encapsulation of the flow. On the other hand, it can also forward a flow back to the next proxy in the same or different service forwarder. The forwarding path is assigned by the controller.

3.3.2. Security Function Proxy. The security function proxy plays an important role in this architecture. Its main function is to support compatibility to an existing traditional (SFC-unaware) security function, because the routing way of the proposed architecture is completely different from the traditional one. Additionally, with the security function proxy, a stateless security function can be converted into a stateful one.

Apparently, the traditional security function cannot recognize a NSH encapsulated flow, because it is designed based on IP network infrastructures. We need a NSH-IP translator to translate the flows received from the service forwarder, which will be processed by a traditional security function. The security function proxy can be regarded as the required translator. It is deployed between a service forwarder and the corresponding security functions. Moreover, similar to the service forwarder, the proxy is also a bidirectional entity, which can translate NSH encapsulated flows into IP encapsulated flows and vice versa. Therefore, the security function proxy includes two kinds of interfaces: the forwarding interfaces and the service interfaces. The former achieves the translation between NSH and IP encapsulation, while the latter is used to communicate with the attached service function.

The other significant function of the proxy is to convert a stateless security function into a stateful one. Each proxy maintains a specific service state list for its attached function and updates it in time. The service state list is shown in Table 1, which includes three columns: *flow ID*, *matching info,* and *service state*. The *flow ID* is an identifier of the flow. To simplify it, we use the common 5-tuple of the flow as the *matching info*. The *service state* records the real state of security function, which can help proxy operate on the flow in the future. When the proxy receives the first packet of a new flow, it redirects the packet to the specified security

Table 1: Service state list.

Flow ID	Matching info	Service state
1	(src_ip1, dst_ip1, protocol1, src_port1, dst_port1)	STATE1
2	(src_ip2, dst_ip2, protocol2, src_port2, dst_port2)	STATE2

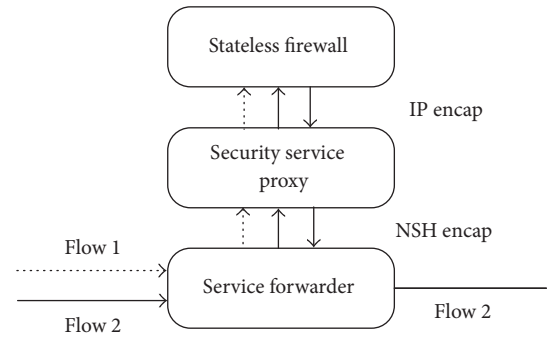


Figure 3: The workflow of a firewall with the security service proxy.

function and adds the identification information of the flow to the service state list, including the flow ID (we use the SPI of this flow) and the matching info of the flow. Once the proxy gets the return flow or the processing result from the security function, it will record the service state of the flow. Furthermore, there is a timer for the proxy to count the return time or delivery time of the flow. If the threshold time is met, the service state of this flow will be written as "*deny*" automatically. Then, according to the service state, the proxy can process following packets of the flow independently to relieve the pressure of the security function.

3.3.3. Security Function. The security functions are the key functional entities in a security service chain. In traditional mobile network architecture, the security services are integrated with other access services. However, in the proposed architecture, the required security services are deployed on mobile-edge network and implemented by the security software. There are two kinds of security functions: the SFC-unaware security functions and the SFC-aware ones. Most of the existed security functions are the former, which should cooperate with the security service proxy. It receives the translated flow from the proxy and executes the predefined security rules, back to the service function proxy. The other security functions are designed for NSH encapsulated flows specially. There is no need to prepare extra proxies for SFC-aware security functions if they are stateful.

The workflow of the virtualized security function is illustrated by an example of achieving a stateful firewall. Figure 3 shows how this stateless firewall is served as a stateful and SFC-aware firewall with the proxy.

There are two flows, flow 1 and flow 2, that enter into the given service forwarder at the same time. Once the service forwarder detects the new flows, it will forward them into its corresponding security service proxy.

When the proxy receives the first packet of a new flow, it will deencapsulate the NSH header and record its flow ID

at first. After that, the flow is sent to the attached firewall immediately. At the same time, the proxy writes the 5-tuple information of the flow into its service state list and launches a timer. If the firewall allows this flow pass through according to its predefined rule, the first packet will return to the proxy in time. And then, the proxy will know its state and record the service state as "*allow*." Otherwise, the service state will be written as "*deny*" when time is over. In the example, we can see that flow 1 is denied but flow 2 is allowed.

Once the service state of the flow is filled in, the proxy can process the following packets of the flow without the help of firewall. In other words, the following packets of the flow are not required to be processed by the firewall any more. So, flow 2 will be forwarded to the service forwarder by this proxy next time and flow 1 will be dropped directly. Finally, for flow 2 which is allowed to be forwarded to next service function, the proxy will restore the NSH encapsulation of flow 2 before sending it out.

3.4. Gateway. The gateway in mobile-edge network is similar to P-GW in traditional mobile network. It is regarded as a packet exit of the mobile-edge network and the entrance to the Internet. In other words, any flow that wants to reach the Internet should pass through this gateway. Moreover, because the gateway is the last hop in mobile-edge network (the end of the security service chain), it checks whether the NSH header of the flow is deencapsulated or not.

In a word, the core idea of the proposed architecture is to remove the security services from traditional mobile network and deploy virtualized security functions on a cloud close to the user side. This change allows the network system to meet flexible and scalable mobile security requirements better.

4. Optimization for Security Service Chaining

In the proposed architecture for mobile communication, we present a security service chain to meet flexible user security requirements by decoupling logical security services with the physical resources in the cloud environment. Then, with the known security requirements, we need to generate a corresponding security service chaining to put required security services into a sequence. In this case, the key point is to determine the order of this sequence, which is dependent on many factors. However, it is difficult to find a best deployment solution because many of the optimal objects are in conflict with each other. For example, if we want to minimize the service processing delay, we have to choose a service node with low CPU usage, which is not beneficial to save power of the whole cloud. Therefore, the practical way to address this problem is to find a proper trade-off, which is not only to guarantee the mobile user experience but also to decrease the energy consumption as much as possible.

In this section, we firstly present a graph based model for security function deployment, which is used to find the proper order of required security functions by our proposed algorithm. Due to the performance of a given security service chain influenced by many factors, we discuss how to make a decision with more than one factor at the same time. Finally, we present a security service composition algorithm based on above considerations, which is used to choose a suitable security service chain with most acceptable performance.

4.1. Graph Based Security Service Model. The security service chaining requires a complex and integrated security service, which consists of a sequence of specialized security functions. In order to avoid the interference from others, different security functions are deployed on different virtual machines (VMs) in the cloud. Because the execution of security functions is in order and there is no security function reused in one security service chain, we use graph theory to describe all security functions and their execution sequence. The definitions are as follows.

4.1.1. Security Function Graph. According to the character of security service chaining, a directed acyclic graph is used to describe all service functions and their execution sequence. The security function graph is formalized as $G = (V, E)$, where V is the set of vertices and E is the set of edges.

4.1.2. Vertex. In the security function graph, a vertex represents an available VM in the cloud. Therefore, a virtualized security function can be deployed on this vertex. Besides, there are two special vertices, which are on behalf of the classifier and the gateway of the cloud correspondingly. The classifier vertex acts as the only entry vertex, and the gateway vertex acts as the only exit vertex. To avoid the interference, each VM can only run one security function in a security service chain. Furthermore, there may be more than one security function running on the same VM for different security service chains.

4.1.3. Edge. Each edge in the security function graph represents the connectivity between two required security functions. The edge between two different VMs is directed because the security functions deployed on them are executed in order. The security function in former vertex should be executed before the later one (next hop). If there are several security service chains, some vertices may have one more edge directed to different next hops. They can be used for different security service chains to deploy their security functions on the same vertex. Besides, there are at least one directed edge from the classifier vertex and one directed edge to the gateway vertex in a security function graph.

4.1.4. Security Function Path. Taking the way of security functions composition into account, the security function path can be expressed as a subgraph of the security function graph. Because all security service chains are started from the classifier and ended at the gateway, a security function path is regarded as a path from the classifier to the gateway in the security function graph. It is formalized as $G_{\text{sfp}} = (V_{\text{sfp}}, E_{\text{sfp}})$, where V_{sfp} is a subset of V and E_{sfp} is a subset of E.

4.1.5. Security Service Chain. The security service chain consists of several different security functions, which are deployed on the security function path. A given security

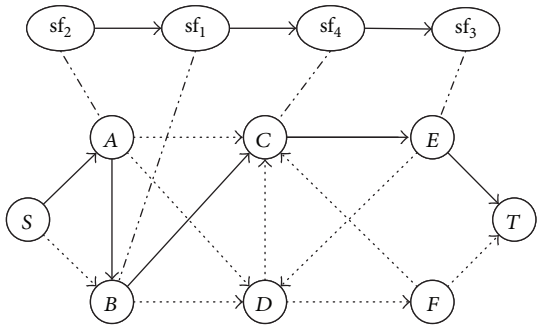


Figure 4: An example of security function graph.

Table 2: Rules for Mamdani fuzzy inference system.

Delay	CPU usage	Acceptance level
Low	High	Highly acceptable
Low	Middle	Acceptable
Low	Low	Acceptable
Middle	High	Acceptable
Middle	Middle	Uncertain
Middle	Low	Unacceptable
High	High	Unacceptable
High	Middle	Unacceptable
High	Low	Completely unacceptable

service chain is formalized as a sequence $\{\mathrm{sf}_1, \mathrm{sf}_2, \ldots, \mathrm{sf}_n\}$ where n is the total number of required security functions. There is a one to one mapping relation between the security functions of a security service chain and the vertices of its corresponding security function path. Furthermore, the size of V_{sfp} should be greater than or equal to $n + 2$ if the number of security functions equals n. Although each security service chain is corresponding to a unique security function path, there may be some vertices and edges reused in the security function graph.

In terms of the graph based security service model, we can formalize a security service chain and its corresponding security function path. The order of security functions can be regarded as match between them, which is the foundation of our proposed algorithm. For example, Figure 4 shows a security function graph $G' = (V', E')$, where $V' = \{S, A, B, C, D, E, F, T\}$. The vertices S and T represent the classifier and the gateway, respectively. The other vertices are normal VMs in this graph, on which we can deploy required security functions. E' is composed of directed lines. We mark a specific security function path with full lines, while the other dotted lines represent other existed security function paths in this graph. This security function path can be described as $G'_{\mathrm{sfp}} = (V'_{\mathrm{sfp}}, E'_{\mathrm{sfp}})$, where $V'_{\mathrm{sfp}} = \{S, A, B, C, E, T\}$. We can deploy an example security service chain $\{\mathrm{sf}_2, \mathrm{sf}_1, \mathrm{sf}_4, \mathrm{sf}_3\}$ on the vertices A, B, C, E in order, and there is a match between the functions and their locations.

4.2. Constrained Optimization for Security Services. In order to meet the user security requirements as much as possible, we choose the required security functions to create the security service chain. However, the order of these functions is supposed to be optimized, which is effected by many extra factors.

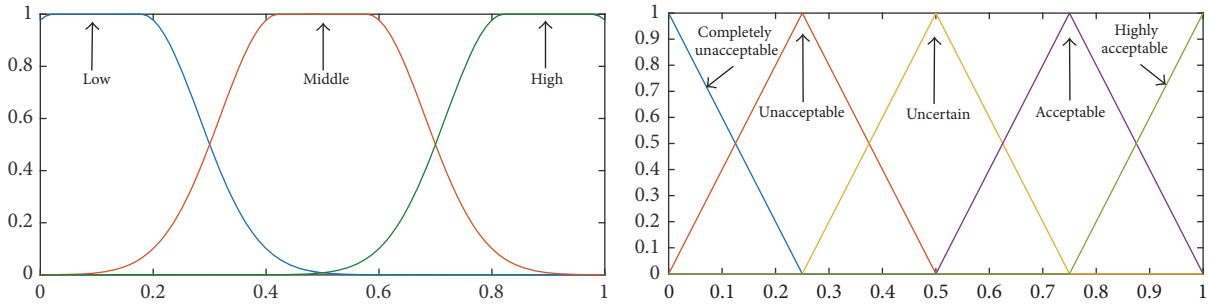
To solve this multiobjective optimization, we should assign proper weight to these objectives. Traditionally, we make the decision by our experience or experts' advices. But this way is coarse-grained because we have to use precise numbers to describe evaluations with ambiguous and semantic words. Thus, we use a fuzzy inference system to describe their weight, which uses fuzzy set theory to map input to output with defined rules [24]. There are two popular types of fuzzy inference system, *Mamdani* [25] and *Takagi-Sugeno*. We choose the Mamdani with centroid defuzzification because it is closer to the human thinking. The input is decided by the number of influential factors, but the output is unique. The fuzzy rules are set by experience or expert advices.

In this paper, we consider two typical aspects, the user experience and system energy consumption. In terms of mobile user experience, the most important factor is delay, including transmission delay and function processing delays. The former is determined when the security function path is given, while the latter is decided by the order sequence. On the other hand, we choose the CPU usage of each VM to evaluate its energy consumption. Therefore, two main optimal objectives are processing delay and CPU usage. At best, we want to choose a composition of security functions with the lowest processing delay and the highest CPU usage.

For our fuzzy inference system, the inputs are processing delay and CPU usage, which are classified in three levels: *low*, *middle,* and *high*. The membership functions of both two inputs are the same shown in Figure 5(a). The only one output is the acceptance level, which is classified in five levels and shown in Figure 5(b). Both input value and output value are normalized numbers. In this case, the output value "zero" in output means the result is completely unacceptable, whereas the value "one" means it is highly acceptable. The rules are as shown in Table 2, which are set by our experience. For example, the first rule means if the processing delay is low and the CPU usage is high, it is regarded as a highly acceptable result.

As illustrated in Figure 6, the proposed fuzzy inference system is able to describe the acceptable level of the output in terms of the CPU usage and the processing delay. In other words, this fuzzy inference system describes a more reasonable weight of these influence factors.

4.3. FIS Based Security Service Composition. For a given security function path $G_{\mathrm{sfp}} = (V_{\mathrm{sfp}}, E_{\mathrm{sfp}})$, there are x candidate VM nodes $\{\mathrm{vm}_1, \mathrm{vm}_2, \ldots, \mathrm{vm}_x\}$ which can employ required security functions $\{\mathrm{sf}_1, \mathrm{sf}_2, \ldots, \mathrm{sf}_x\}$. Taking many different influential factors into account, such as their processing delay and CPU usage, we need to find an appropriate deployment of these security functions. Therefore, based on the



(a) The shape of input membership function (b) The shape of output membership function

FIGURE 5: The input and output membership functions for the purposed fuzzy inference system.

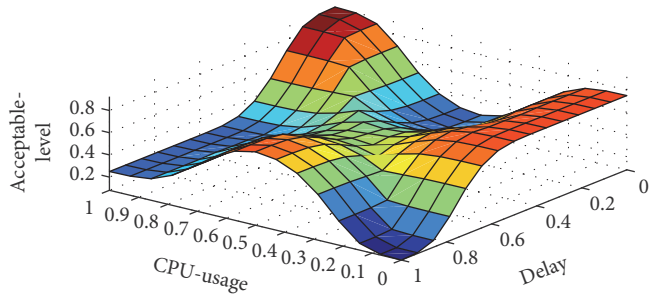


FIGURE 6: The acceptable level with different CPU usage and processing delay.

fuzzy inference system, we propose a new algorithm to find the match between required security functions and their locations (orders). In this way, we assume that the chosen influential factors of each security function are measured previously.

The algorithm to find the most suitable security function composition is presented as follows.

Step 1. Set the influential factors of each security function F_k $(k = 1, 2, \ldots, y)$ for each alternative VM node N_j $(j = 1, 2, \ldots, x)$ with respect to security function S_i $(i = 1, 2, \ldots, x)$ denoted by $R_k = X_{ijk}$.

Step 2. Compute relative membership degrees for the VM nodes and the security functions with different influential factors. For a certain influential factor $(k = u)$, if the bigger influential factor is better, its relative membership degree is shown in

$$r_{ijk}\big|_{k=u} = \frac{x_{ijk}\big|_{k=u} - x_{\min}}{x_{\max} - x_{\min}}, \tag{1}$$

where $x_{\min}$ is the minimum $x_{ijk}|_{k=u}$ and $x_{\max}$ is the maximum $x_{ijk}|_{k=u}$.

Otherwise, its relative membership degree is shown in

$$r_{ijk}\big|_{k=u} = \frac{x_{\max} - x_{ijk}\big|_{k=u}}{x_{\max} - x_{\min}}. \tag{2}$$

Step 3. Set relative membership degrees of the chosen influential factors $(r_{ij1}, r_{ij2}, \ldots, r_{ijy})$ as the input of our fuzzy inference system; then we can get the acceptable level r_{ij} in output.

Step 4. Therefore, the fuzzy relation matrix is shown in

$$R = \left[r_{ij}\right]_{x*x}. \tag{3}$$

Step 5. Use the Hungarian method [26] to find the independent zero element in R, which represents the matching relation between the security function and its corresponding VM.

Step 6. Deploy these security functions in order.

5. Implementation and Evaluation

5.1. Prototype Implementation. For a typical scenario of 3GPP mobile LTE network, our proposed mobile-edge architecture should include an UE, some necessary access services for LTE, P-GW, required security functions, and the interface to the Internet. However, to make it clear, we only implement the proposed mobile-edge cloud in our prototype.

As illustrated in Figure 7, the prototype consists of a classifier, a gateway, four security functions with their corresponding proxies and service forwarders. The classifier is responsible to create a flow and assign an appropriate security

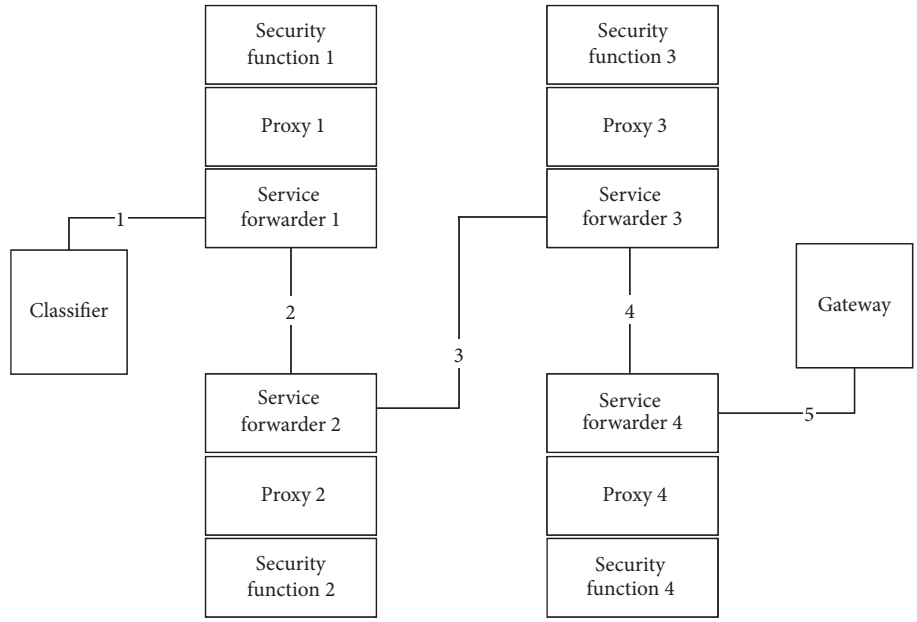


Figure 7: The test bed of mobile-edge cloud.

service chain. The flow will pass through the security functions 1–4 in order. Finally, the gateway receives this flow. There is a security function proxy between each service forwarder and its security function to translate the flow from IP to NSH encapsulation and vice versa. Each of them runs on a VM with dual-core CPU, 2 GB memories and several NICs, which is created by Kernel-based Virtual Machine (KVM) in OpenStack.

The classifier and service forwarders are implemented by OpenVSwitch [27]. OpenVSwitch is a software switch, which is able to forward specific flow based on predefined rules. In our prototype, we use the SDN controller POX [28] to control the behavior of the OpenVSwitch. To achieve SFC routing, service forwarders are equipped with Nshkmod [29]. Nshkmod is an open source Linux kernel module implementation of Network Service Header, which is used to encapsulate and deencapsulate NSH for service flows. The implementation of the security function proxy is also based on OpenVSwitch and Nshkmod, in order to remove/insert NSH encapsulation of packets and redirect them to the next service forwarder or the receiver.

To achieve the standard SFC routing between two different VMs, each virtual NSH interface (named nshx), created by Nshkmod, is bound up with a NIC (named ethx) in an ovs net-bridge. The ovs net-bridge has the ability to forward packets from a virtual NSH interface to its related NIC in order to encapsulate it by NSH header and vice versa. Additionally, the data transmission between two virtual NSH interfaces is accomplished by a special tunnel in Linux kernel. In this way, a service forwarder can communicate with its corresponding security function proxy. The security function proxy for a service forwarder can send/receive data to/from its corresponding security function in a traditional routing way (IP routing).

Table 3: Processing delay of the required functions on different VMS.

	VM1	VM2	VM3	VM4
Service 1	15	12.5	14.6	13.8
Service 2	10.4	10.8	11.5	11.8
Service 3	8.7	9.1	8.2	8.9
Service 4	5.8	8.1	7.3	6.4

Table 4: CPU usage of different VMS with the required functions.

	VM1	VM2	VM3	VM4
Service 1	83.1	85.2	84.7	84.4
Service 2	25.6	24.2	25.5	27.1
Service 3	22.1	24.7	23.9	23.3
Service 4	68.5	72.3	71.9	70.8

Considering the most common security services in mobile network, we choose the firewall, network address translator (NAT), Deep Packet Inspection (DPI), and load balancer (LB) as an example. The firewall and NAT are both implemented by iptables [30], which is integrated into the Linux kernel. The DPI is implemented by nDPI [31], a famous open source software. The LB is implemented by Linux Virtual Server (LVS) [32], a popular technology of load balance server.

We assume that security function path is decided as shown in Figure 7. We choose two important influence factors mentioned in Section 4, the processing delay of security functions, and the CPU usage of their corresponding VMs. Based on the prototype, we measure these parameters for each security function on four decided VMs independently. The measurement result of processing delay is shown in Table 3 and the CPU usage is shown in Table 4.

5.2. Contrast Algorithm for Security Service Composition. In order to evaluate the performance of our proposed algorithm, a contrast algorithm is presented to solve the problem of security service composition. The simple additive weighting (SAW) [33] is widely used for service composition, which transforms a multiobjective optimization into single-objective optimization. However, the SAW algorithm is traditionally used without fuzzy theory. To make it reasonable to evaluate our fuzzy algorithm, we present a fuzzy SAW based algorithm, which combines the SAW algorithm with fuzzy relation theory.

The problem of security service composition and its conditions are the same to Section 4.3. The detailed algorithm is presented as follows.

Step 1. Set the performance evaluations of each security function P_k $(k = 1, 2, \ldots, y)$ for each alternative VM node N_j $(j = 1, 2, \ldots, x)$ with respect to security function S_i $(i = 1, 2, \ldots, x)$ denoted by $R_k = X_{ijk}$.

Step 2. Compute relative membership degrees for the VM nodes and the security functions with different performance evaluations. For a certain performance evaluation $(k = u)$, if the bigger performance evaluation is better, its relative membership degree is shown in

$$r_{ijk}\big|_{k=u} = \frac{x_{ijk}\big|_{k=u} - x_{\min}}{x_{\max} - x_{\min}}, \tag{4}$$

where $x_{\min}$ is the minimum $x_{ijk}|_{k=u}$ and $x_{\max}$ is the maximum $x_{ijk}|_{k=u}$.

Otherwise, its relative membership degree is shown in

$$r_{ijk}\big|_{k=u} = \frac{x_{\max} - x_{ijk}\big|_{k=u}}{x_{\max} - x_{\min}}. \tag{5}$$

Step 3. Set the fuzzy weight $W = (w_1, w_2, \ldots, w_y)$, where $\sum w_k = 1$ $(k = 1, 2, \ldots, y)$ in terms of our experience or experts' advices.

Step 4. According to the fuzzy weight W, compute the aggregate fuzzy performance evaluations, which is shown in

$$r_{ij} = \sum w_k \cdot r_{ijk}. \tag{6}$$

Step 5. Therefore, the fuzzy relation matrix is shown in

$$R = \left[r_{ij}\right]_{x*x}. \tag{7}$$

Step 6. Use the Hungarian method to find the independent zero element in R, which represents the matching relation between the security function and its corresponding VM.

Step 7. Deploy these security functions in order.

5.3. Evaluation Method and Criteria. A single-objective optimization algorithm can be easily evaluated by calculating the best value, because its target is to find only one extreme value of the given problem. However, it is very hard to evaluate a multiobjective optimization algorithm. Usually, a multiobjective optimization problem is influenced by some factors which are in conflict with each other. It is impossible to find the solution to satisfy every optimal objective in one time. Therefore, for a multiobjective optimization, there exist a lot of Pareto optimal solutions. The Pareto optimal solution cannot be improved in any objective for this problem if we do not degrade the other objectives. The Pareto front is a set of all the Pareto optimal outcomes, which can be used to describe the best solution of a multiobjective optimization problem.

Therefore, it is necessary to find the Pareto front before we evaluate the proposed algorithm. Because the Pareto front of the proposed security service composition problem is unknown, it is supposed to choose an existing algorithm to estimate the Pareto front. The genetic algorithm is one of the most popular algorithms for multiobjective optimization, which can help us obtain the Pareto front. In this paper, we use NSGA-II [34] algorithm, which is a kind of improved genetic algorithm, to estimate the Pareto front. Because the Pareto front achieved by this algorithm is finite and dynamically changed every run time, we run this algorithm for 10 times under the same situation to collect the Pareto points as much as possible.

The NSGA-II algorithm we used is decided by three main parameters: the population size, the maximum number of generation, and the fitness function for our problem. There is no comprehensive solution for setting the population size and the maximum number of generations. For example, with a smaller population, it is fast to find the solution but may lead to local optimum, while a larger population may cause low efficiency. The decision of maximum number of generations is similar to the population size. Therefore, we evaluate the algorithm under several situations with different settings of the population size and the maximum number of generations.

On the other hand, it is important to choose a suitable fitness function for our problem, which has a great effect on the rate of convergence of the algorithm. In terms of the model proposed in Section 4.3, we assume that there are k influential factors of each security function and j alternative VM node with its respective security function where $j \in \{1, 2, \ldots, x\}$ and $k \in \{1, 2, \ldots, y\}$. The value of the kth influential factors on each VM is denoted as q_{kj}. Moreover, these factors can be classified into two categories: the positive factor Q^+ and the negative factor Q^-. For the former, larger value means the better, while for the latter, small value means the better. Then, the fitness functions f_k are shown in

$$f_k = \begin{cases} \sum_{j=1}^{x} q_{kj}, & \text{if } q_{kj} \in Q^-, \\ -\sum_{j=1}^{x} q_{kj}, & \text{if } q_{kj} \in Q^+. \end{cases} \tag{8}$$

According to the calculation outcomes, we can obtain a large number of Pareto optimal points. Each point is a possible optimal solution for our problem, which is regarded as the estimated Pareto optimal point. With the estimated Pareto front, the next step is to evaluate our fuzzy based optimization

Table 5: Relative membership degree of processing delay.

	VM1	VM2	VM3	VM4
Function 1	0	0.27	0.04	0.13
Function 2	0.5	0.46	0.38	0.35
Function 3	0.68	0.64	0.74	0.66
Function 4	1	0.75	0.84	0.93

Table 6: Relative membership degree of CPU usage.

	VM1	VM2	VM3	VM4
Function 1	0.97	0	0.99	0.99
Function 2	0.06	0.03	0.05	0.02
Function 3	0	0.04	0.03	0.02
Function 4	0.73	0.80	0.79	0.77

algorithm. The reasonable way to evaluate a multiobjective optimization algorithm is to make a comparison between the calculated result and the Pareto front. There is a widely used indicator to describe the difference between the estimated Pareto front and the result calculated by our algorithm, called IGD. IGD represents the distance from the result achieved by our algorithm to the estimated Pareto front. It is obvious that a smaller value is better, because it is closer to the Pareto front. IGD can be calculated by

$$\mathrm{IGD} = \frac{1}{n}\sqrt{\sum_{i=1}^{n} |x_i - x|^2}, \tag{9}$$

where n is the total number of Pareto optimal points, x is the influential vector of the result, and x_i is the influential factor vector of the ith Pareto optimal point.

Therefore, we can use NSGA-II algorithm to find the Pareto front under a certain situation at first. Then, we calculate the IGD value of a given security service composition algorithm according to the achieved Pareto front. Finally, we evaluate the performances of different algorithms by comparing the IGD values between them, and the algorithm with smaller IGD values is better.

5.4. Experimental Results. According to the proposed algorithm, we calculate the corresponding relative membership degrees for each measurement firstly, as shown in Tables 5 and 6, respectively.

Then, we use the fuzzy inference system presented in Section 4 to make a trade-off decision between these two factors. Set the relative membership degrees of processing delay and CPU usage as the input; then we get the fuzzy relation matrix R.

$$R_{\mathrm{FIS}} = \begin{bmatrix} 0.8975 & 0.7436 & 0.8946 & 0.8946 \\ 0.5714 & 0.6329 & 0.7153 & 0.7163 \\ 0.1315 & 0.1865 & 0.1255 & 0.1522 \\ 0.2557 & 0.3166 & 0.2940 & 0.2654 \end{bmatrix}. \tag{10}$$

Table 7: Matching relation between services and VMS.

	VM1	VM2	VM3	VM4
Function 1	*	0	*	*
Function 2	0	*	*	*
Function 3	*	*	0	*
Function 4	*	*	*	0

Table 8: Reference matching relation between services and VMS.

	VM1	VM2	VM3	VM4
Function 1	*	0	*	*
Function 2	*	*	*	0
Function 3	0	*	*	*
Function 4	*	*	0	*

Finally, we use Hungarian method to find out the matching relation between security functions and VMs, which is marked by "0" in Table 7.

According to Table 7, we should deploy the security function 1 on VM2, function 2 on VM1, function 3 on VM3, and service 4 on VM4. Therefore, the service deployment order of the given security service chain is function 2, function 1, function 3, and function 4. The total processing delay in this way equals 37.5 and its corresponding total CPU usage equals 205.5.

To evaluate our algorithm, the same problem is also calculated by the fuzzy SAW based algorithm presented in Section 5.2. We set the reference fuzzy weight $W_{\mathrm{SAW}} = (W_{\mathrm{delay}}, W_{\mathrm{CPU}}) = (0.6, 0.4)$. It means the influence of processing delay is a little larger than that of CPU usage, which is similar to rules we defined for our fuzzy inference system. Different from the algorithm, we use a precise number to describe our subjective feeling in this way. With this empirical fuzzy weight, we use (6) to compute the aggregate membership degree, which is shown in (11).

$$r_{ij} = \sum w_k \cdot r_{ijk} = 0.6r_{ij1} + 0.4r_{ij2}. \tag{11}$$

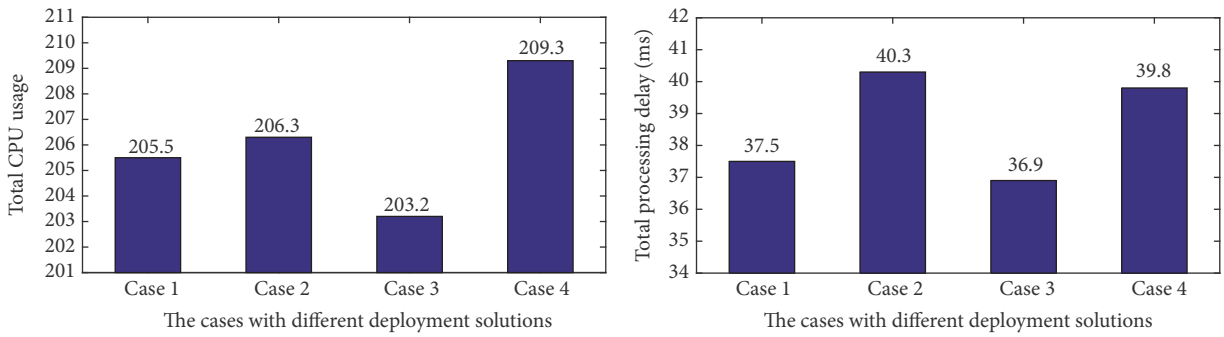
So, the reference fuzzy relation matrix is

$$R_{\mathrm{SAW}} = \left[r_{ij}\right]_{4*4} = \begin{bmatrix} 0.388 & 0.162 & 0.420 & 0.474 \\ 0.324 & 0.288 & 0.248 & 0.242 \\ 0.408 & 0.400 & 0.456 & 0.404 \\ 0.532 & 0.500 & 0.518 & 0.531 \end{bmatrix}. \tag{12}$$

Similarly, we use Hungarian method to find out the reference matching relation between security functions and VMs, which is marked by the "0" in Table 8.

According to Table 8, we should deploy the security function 1 on VM2, function 2 on VM4, function 3 on VM1, and service 4 on VM3. Therefore, the service deployment order of the given security service chain is function 3, function 1, function 4, and function 2. The total processing delay in this way equals 40.3 and its corresponding total CPU usage equals 206.3.

To evaluate our algorithm objectively, we also calculate two extreme cases in this scenario, which are the shortest



(a) Total processing delay different cases

(b) Total CPU usage for different cases

Figure 8: Comparisons between total processing delay and total CPU usage for different cases.

total processing delay case and the maximum total CPU usage case. In the former case, the total processing delay equals 36.9 and its corresponding total CPU usage equals 203.2. While the total processing delay of the latter case equals 39.8 and its total CPU usage equals 209.3.

For convenience, we mark the four cases as cases 1–4: case 1 represents the deployment with FIS based algorithm, case 2 represents the deployment with fuzzy SAW based algorithm, and case 3 and case 4 represent shortest processing delay case and maximum CPU usage case, respectively.

To make it clear, we discuss the two factors, total processing delay and total CPU usage equal independently. Figure 8(a) shows the total processing delays for different cases. We can see case 1 is very similar to case 3, which represents the minimal total processing delay. It is obvious that the deployment with our algorithm can obtain lower processing delay, which is close to our object. While case 2 and case 4 are much higher than them, particularly case 2 is even higher than case 4. Maybe case 2 is the worst case in this scenario because it obtains neither the shortest processing delay nor the highest CPU usage. Although the subjective feeling of contrast algorithm seems like FIS based algorithm, the result shows that it is extremely different. The reason is the criteria for these factors are semantic and complexity. It is difficult to describe the criteria with only one precise number. The result also proves that our fuzzy inference system is useful to solve this problem.

On the other hand, Figure 8(b) shows the total CPU usage for different cases. Although there is no other case similar to case 4, which represents the maximum total CPU usage, case 1 and case 2 are much higher than case 3. Case 3 may be the lowest CPU usage case in this scenario at the cost of its shortest processing delay. Case 2 is very close to case 1, so both of them can be the deployment candidate with higher CPU usage. However, according to the above analysis about processing delay, case 2 is the worst solution. Therefore, the most proper deployment is case 1. In other words, our algorithm provides a best trade-off between these two factors. It proves that the algorithm we proposed is useful to create an appropriate security service chain with multiple influence factors.

5.5. Performance Analysis. In this section, we firstly estimate the Pareto front of our problem by the NSGA-II algorithm, because there are four VMs needed to deploy security functions in the above scenario and two influential factors to be considered. So, in terms of (4), the fitness functions are supposed to be

$$\begin{aligned} f_1 &= q_{11} + q_{12} + q_{13} + q_{14}, \\ f_2 &= q_{21} + q_{22} + q_{23} + q_{24}. \end{aligned} \tag{13}$$

In order to make it closer to the real scenario, we set these decision variables of two fitness functions (objective functions) based on the measurements in Tables 3 and 4. Thus, the range of these decision variables is as follows:

$$\begin{aligned} q_{11} &\in (12, 16), \\ q_{12} &\in (10, 12), \\ q_{13} &\in (8, 10), \\ q_{14} &\in (5, 9), \\ q_{21} &\in (83, 86), \\ q_{22} &\in (24, 28), \\ q_{23} &\in (22, 25), \\ q_{24} &\in (68, 73). \end{aligned} \tag{14}$$

Because the population size and the maximum number of generations are difficult to determine in this scenario, we use four different pairs of values for the population size and the maximum number of generation to evaluate the performance of algorithms. These pairs of values are set as (20, 100), (50, 200), (50, 100), and (100, 200) where the former represents its population size and the latter represents its maximum number of generation.

After we obtain all the Pareto optimal points for all the situations, we calculate their IGD values with (8). In this paper, we calculate the IGD values for both FIS based algorithm and fuzzy SAW based algorithm under above four

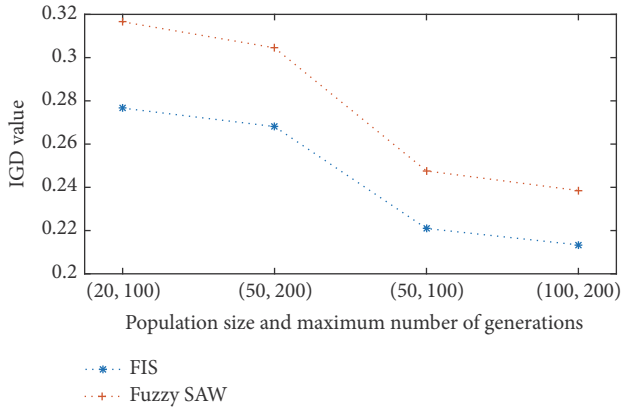


Figure 9: Comparisons of the IGD values of different security service composition algorithms.

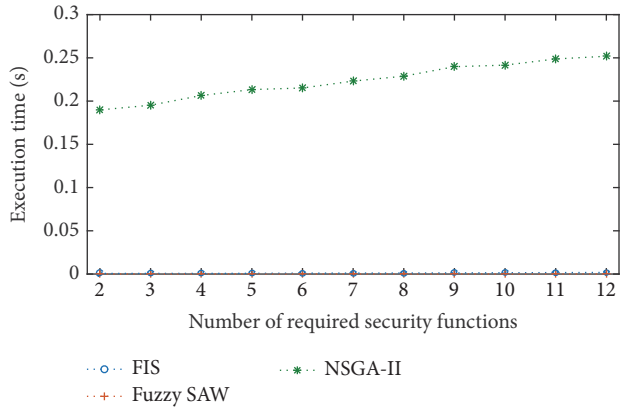


Figure 10: Comparing the execution time of different security service composition algorithms.

different situations. As shown in Figure 9, the dotted line with plus signs represents IGD values of proposed FIS based algorithm, while the dotted line with asterisk represents that of the fuzzy SAW based algorithm. Each line consisted of four points and each point represents one of the situations. For example, the first point in the dotted line with plus signs means that IGD value of FIS based algorithm is calculated with population size 20 and maximum number of generations 100.

It is obvious that the dotted line with asterisk is always lower than the dotted line with plus signs. Additionally, there are downturns for both of the dotted lines shown in Figure 9. The probable reason is that the population size is not large enough to find adequate number of Pareto optimal points by NSGA-II algorithm. Moreover, too much generations with gene mutation also make the IGD values larger because the IGD value means the stability of chosen samples. In general, IGD values of FIS based algorithm are always smaller than that of fuzzy SAW based algorithm under each situation. It indicates that the result achieved by our proposed algorithm is closer to the best solution (Pareto front). Thus, the solutions of FIS based algorithm are better than fuzzy SAW algorithm.

Besides, we also make a comparison about the execution time of different algorithms shown in Figure 10. The dotted line with circles represents the execution time of proposed FIS based algorithm, the dotted line with plus signs represents that of comparison algorithm, and the dotted line with asterisk represents that of the NSGA-II algorithm. In this experiment, we calculate the execution time of each algorithm for different number of required security functions in a security service chain, which is from 2 to 12. Evidently, the execution time of NSGA-II is always much larger than other algorithms. With the number of required security functions increasing, the execution time is also increasing. However, according to our achieved results, the performance of FIS based algorithm is similar to fuzzy SAW based algorithm, and both the execution times of FIS based and fuzzy SAW based algorithms are lower than 1 ms. Thus, it is acceptable for security service chaining to choose either the FIS based algorithm or the fuzzy SAW based one in terms of the execution time.

Based on the above analysis, the overall performance of FIS based algorithm is better than that of the fuzzy SAW based algorithm.

6. Conclusion

In this paper, we present a new architecture for sustainable MEC with the idea of security service chaining. The proposed service chaining can meet more diversity and flexible mobile user requirements. Moreover, we present a security function proxy, which provides the compatibility to traditional security functions in the new architecture. We also propose an algorithm to find a proper way to composite required security functions with multiple influential factors. In detail, the security service chain is described by a graph model, and a fuzzy inference system is adopted to find the suitable order of the required security functions. Finally, with an implemented prototype, we calculate and analyze the performance of our proposed algorithm by comparing a widely used SAW algorithm. The experimental results obviously prove the proposed FIS based algorithm is better than the contrast one with a common evaluation criterion.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by National High Technology of China ("863 program") under Grant no. 2015AA015702, National Basic Research Program of China ("973 program") under Grant no. 2013CB329101, NSFC of China under Grant no. 61271202 and no. 61602030, and NSAF under Grant no. U1530118.

References

[1] Cisco, *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015–2020*, CISCO, 2016.

[2] M. Patel, B. Naughton, C. Chan et al., "Mobile-edge computing—introductory technical white paper," Tech. Rep. 1, The European Telecommunications Standards Institute, 2014.

[3] C. E. Weng, V. Sharma, H. C. Chen, and C. H. Mao, "PEER: proximity-based energy-efficient routing algorithm for wireless sensor networks," *Journal of Internet Services and Information Security*, vol. 6, no. 1, pp. 47–56, 2016.

[4] S. Aram, I. Khosa, and E. Pasero, "Conserving energy through neural prediction of sensed data," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 6, no. 1, pp. 74–97, 2015.

[5] G. Orsini, D. Bade, and W. Lamersdorf, "Computing at the mobile edge: designing elastic android applications for computation offloading," in *Proceedings of the 8th IFIP Wireless and Mobile Networking Conference (WMNC '15)*, pp. 112–119, Munich, Germany, 2015.

[6] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2015.

[7] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, 2015.

[8] M. T. Beck, S. Feld, A. Fichtner, C. Linnhoff-Popien, and T. Schimper, "ME-VoLTE: network functions for energy-efficient video transcoding at the mobile edge," in *Proceedings of the 18th International Conference on Intelligence in Next Generation Networks (ICIN '15)*, Paris, France, February 2015.

[9] E. J. Halpern and E. C. Pignataro, "Service function chaining (sfc) architecture," Tech. Rep. RFC 7665, Internet Engineering Task Force (IETF), 2015.

[10] B. Rashidi and C. Fung, "A survey of android security threats and defenses," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 6, no. 3, pp. 3–35, 2015.

[11] P. Quinn and U. Elzur, "Network service header," IETF Internet-Draft, 2016, https://datatracker.ietf.org/doc/draft-ietf-sfc-nsh/04/.

[12] F. Callegati, W. Cerroni, C. Contoli, and G. Santandrea, "Dynamic chaining of virtual network functions in cloud-based edge networks," in *Proceedings of the 1st IEEE Conference on Network Softwarization (NETSOFT '15)*, IEEE, London, UK, April 2015.

[13] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets-IX '10)*, pp. 1–6, Monterey, Calif, USA, 2010.

[14] A. Gember, A. Krishnamurthy, S. S. John et al., "Stratos: a network-aware orchestration layer for virtual middleboxes in clouds," https://arxiv.org/abs/1305.0209.

[15] S. K. Fayazbakhsh, L. Chiang, V. Sekar, M. Yu, and J. C. Mogul, "Enforcing network-wide policies in the presence of dynamic middlebox actions using flowtags," in *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation (NSDI '14)*, pp. 543–546, Seattle, Wash, USA, 2014.

[16] Z. A. Qazi, C.-C. Tu, L. Chiang, R. Miao, V. Sekar, and M. Yu, "SIMPLE-fying middlebox policy enforcement using SDN," in *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '13)*, pp. 27–38, Hong Kong, China, August 2013.

[17] C. Yu and M.-T. Yuan, "Web service composition using graph model," in *Proceedings of the 2nd IEEE International Conference on Information Management and Engineering (ICIME '10)*, pp. 656–660, IEEE, Chengdu, China, April 2010.

[18] P. J. Sun, P. C. Zhang, W. R. Li, X. J. Guo, and J. Feng, "Sky-MCSP-R: an efficient graph-based Web service composition approach," in *Proceedings of the 20th Asia-Pacific Software Engineering Conference (APSEC '13)*, pp. 589–594, IEEE, Bangkok, Thailand, December 2013.

[19] C.-Q. Jiang, W. Du, and C.-C. Yi, "A method of web service composition based on bipartite graph optimal matching and QoS," in *Proceedings of the International Conference on Internet Technology and Applications (ITAP '10)*, pp. 1–5, IEEE, Wuhan, China, August 2010.

[20] S. A. da Silva, H. Ma, and M. J. Zhang, "A graph-based Particle Swarm Optimisation approach to QoS-aware web service composition and selection," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '14)*, pp. 3127–3134, Beijing, China, July 2014.

[21] N. Kashyap and K. Tyagi, "Dynamic composition of web services based on Qos parameters using fuzzy logic," in *Proceedings of the 2nd International Conference on Advances in Computer Engineering and Applications (ICACEA '15)*, pp. 778–782, March 2015.

[22] M. Bakhshi, F. Mardukhi, and N. Nematbakhsh, "A fuzzy-based approach for selecting the optimal composition of services according to user preferences," in *Proceedings of the 2nd International Conference on Computer and Automation Engineering (ICCAE '10)*, vol. 5, February 2010.

[23] A. V. Dastjerdi and R. Buyya, "Compatibility-aware cloud service composition under fuzzy preferences of users," *IEEE Transactions on Cloud Computing*, vol. 2, no. 1, pp. 1–13, 2014.

[24] L. A. Zadeh, "Fuzzy logic = computing with words," *IEEE Transactions on Fuzzy Systems*, vol. 4, no. 2, pp. 103–111, 1996.

[25] E. H. Mamdani and S. Assilian, "Experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, vol. 7, no. 1, pp. 1–13, 1975.

[26] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1955.

[27] B. Pfaff, J. Pettit, T. Koponen et al., "The design and implementation of open vswitch," in *Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation (NSDI '15)*, pp. 117–130, Oakland, Calif, USA, 2015.

[28] N. Repo and contributors, The pox controller, 2012–2014, https://github.com/noxrepo/pox.

[29] R. Nakamura, Nshkmod, 2015-2016, https://github.com/upa/nshkmod.

[30] Netfilter Core Team and Contributors, iptables, 1999–2014, http://www.netfilter.org/projects/iptables/index.html.

[31] The Ntop Team, nDPI, 2016, http://www.ntop.org/.

[32] Linux Virtual Server, 2016, http://www.linuxvirtualserver.org/.

[33] S.-J. Chen and C.-L. Hwang, *Fuzzy multiple attribute decision making*, vol. 375 of *Lecture Notes in Economics and Mathematical Systems*, Springer-Verlag, Berlin, Germany, 1992.

[34] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.