Scalable Computing:
Practice and Experience

Scientific International Journal
for Parallel and Distributed Computing

ISSN: 1895-1767

Universitatea de Vest din Timişoara

Volume 20(3) September 2019

SUBSCRIPTION INFORMATION: please visit http://www.scpe.org

Scalable Computing: Practice and Experience

Volume 20, Number 3, September 2019

TABLE OF CONTENTS

DOI 10.12694/scpe.v20i3.1323
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 3, pp. iii–iv. http://www.scpe.org

INTRODUCTION TO THE SPECIAL ISSUE ON IOT CLOUD SOLUTIONS FOR SOCIETAL APPLICATIONS

Addressing societal problems such as air pollution, water contamination, corruption, healthcare management, agricultural assistance, and so forth has increased in the recent past using several highend technologies, including IoT cloud.

In fact, innovations have become the key factor of economic growth in several developing and developed countries. For instance, Atal Innovation Mission of India has promoted innovations that addresses the needs of the society through IoT cloud based technologies at AIC-IIITKottayam; TUM-Germany and several top ranked universities across the globe have driven the research or product developments targeting the benefits of the society energy, healthcare, agriculture, economic health, and so forth.

Existing IoT based societal applications require a large volume of data for the analysis; a secure environment for handling data (either in the cloud or edge environments); and, a diligent planning while handling heterogeneous devices on federated cloud environments. In fact, scalability of cloud resources is the backbone for IoT cloud environments, while edge computing is required for short response times. Many IoT applications are based on automatic decision making applying machine learning on huge data sets and thus, these techniques influenced a large group of researchers in the IoT cloud domain.

This special issue on IoT Cloud Solutions for Societal Applications discusses security aspects and a decision making in IoT applications. Syed et al discuss IoT security requirements, challenges, and in the context of smart cities, smart health, smart building, smart transport, and smart industry applications; Naveen et al have carried out a theoretical evaluation using spacio-temporal distance networks in a deep learning network in order to perform the action recognition tasks in IoT environments.

Shajulin Benedict, Indian Institute of Information Technology Kottayam, Kerala, India
Michael Gerndt, Technische Universitaet Muenchen, Germany

DOI 10.12694/scpe.v20i3.1544
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 3, pp. 457–484. http://www.scpe.org

ASSESSING THE SERVICES, SECURITY THREATS, CHALLENGES AND SOLUTIONS IN THE INTERNET OF THINGS

SYED RAMEEM ZAHRA AND MOHAMMAD AHSAN CHISHTI*

Abstract. The purpose of this paper is to chalk out the criticality of the most important pillar of the Internet of Things (IoT), i.e., Security and Privacy (S&P). IoT has seen its journey from implausible and impossible to sustainable and tenable. Its rate of expansion into various grounds from agriculture to sports; personal health to intelligent traffic detection; waste management to smart homes is astonishing, dramatic, and unforeseen. With such vast adaptability and functionality, its security remains the biggest concern because, in contrast to the traditional networks, IoT faces huge vulnerabilities, some of which are inherent and others explicit. The existing security solutions cannot be implemented in IoT because of its unique characteristics. Therefore, there is a dire need to develop novel security procedures befitting IoT. This paper spots the features that are peculiar to IoT and concurrently analyzes the security threats, and challenges they pose. This work also provides a glimpse of the major IoT implementations with their particular security requirements and challenges. Moreover, this paper critically evaluates the proposed countermeasures to security attacks on different features and why they cannot be used in IoT environments. Also, it is found that most of the security solutions used in IoT devices are inspired by Wireless Sensor Networks (WSN, but the striking differences among the two make them inadequate in IoT. The security requirements and challenges peculiar to various IoT services are also identified. To assist the researchers in remaining up-to-date, we for the first time have thoroughly expressed some of the most famous and practical attacks faced across the world in the recent past, how much damage they caused, how many financial losses were faced, etc.

Key words: Internet of Things, Security, Privacy, Vulnerabilities, Wireless Sensor Networks.

AMS subject classifications. 68M14, 68M10

1. Introduction.

Internet of Things (IoT) pilots the automation in an ample number of realms ranging from management of items with trivial importance like thermostats to the management of life-saving medical implants. The application spectrum of IoT runs from monitoring the dampness in crops, to auditing the flow of items through a production line, to remotely observing the patients with interminable illnesses and overseeing their restorative devices. It is to say that the potential application areas of IoT are innumerable and diverse, percolating into all the spheres of individual lives as well as into the enterprises and society as a whole. The European Research Cluster on the Internet of Things (IERC) identifies primal applications of IoT that span numerous domains and describe them as Smart City, Smart Health, Smart Buildings, Smart Transport, and Smart Industry.

As IoT maneuvers past a catchphrase and begins to offer solutions to such a wide range of multi-faceted problems, a clear understanding of 3 of its vital pillars has been achieved [1] a) the foundation of contextual awareness is laid by the blend of sensors and actuators which make the interaction with the environment as well as the transformation of stimulus to data and vice versa possible b) the devices used in IoT are highly constrained in terms of power, bandwidth, processing abilities, memory, and size. Hence in the missions where less latency, consideration to less bandwidth usage and real-time analytics is needed, local edge computing and fog computing become essential c) data exchange between the IoT devices and the local aggregators or cloud happens through low power communication links.

Left out from this picture, and not completely acknowledged yet, is the fourth pillar of IoT: Security and Privacy (S&P). Given that all the vital elements of IoT- people, processes and things work together just to create more data and to extract profitable and relevant information from that data, then how the S&P is dealt with will decide the destiny of IoT i.e. whether there will be a second round of rapid expansion and escalation of IoT or an extreme downfall and debacle.

Recent breaks in S&P are changing the way businesses view this matter because even the tiny IoT devices that have restricted functionality pose serious dangers to the entire security system of the network when their security is compromised. This is because by connecting everything to the internet, IoT creates a huge attack surface for the rogue players and weak points could easily be targeted and compromised to set off an attack and steal sensitive data. Therefore our approach of looking at IoT should be changed, making S&P a vital

*Department of Computer Science Engineering, National Institute of Technology Srinagar, India. (rameemzahra@gmail.com).

requirement at the design phase itself. Also, the major research conducted in the direction of S&P of IoT mainly tries to adapt the security solutions aimed at Wireless Sensor Networks (WSN) and internet to IoT [2]. However, on contemplating the inherent features of IoT and its differences from WSN, we come across a glaring reality which says that IoT challenges take another dimension which is a long way from being anything but difficult to defeat with customary solutions. In essence, the contribution of this paper includes:

1. Identification of basic features of IoT and how they constitute the internal security vulnerabilities of IoT devices.
2. Primal applications of IoT are studied from their security point of view.
3. An exhaustive study of various papers and projects proposed in the realm of IoT applications and security.
4. Examination of various attacks targeting the vulnerabilities of IoT devices and causing huge financial losses.
5. The critical analysis of existing threats and challenges about the identified features.

The rest of the paper is organized as follows: Section 2 gives a background about the intrinsic IoT features which are fundamental to any IoT application. In Section 3, we discuss in detail five important IoT application use cases that are identified by IERC. It also sketches out the security challenges, and requirements of the described IoT applications. The major threats, challenges and the proposed solutions posed in the entire IoT environment by the intrinsic IoT features are discussed in Section 4. It also describes the problems that exist with the given solutions. Section 5 concludes the paper.

2. Inherent IoT Features. Less storage capacity, small battery back-up, and limited compute ability mark the identity of IoT devices. As such, constrained is one of the inherent features of these devices. Apart from being constrained, the uniqueness of IoT devices is marked by features like Interdependence, Heterogeneity, Constrained, Pervasiveness, Unattended, Affinity, and Mobility [3]. These features also represent the critical inherent vulnerabilities of IoT devices and are briefly explained below:

1. **Interdependence:** The root cause of security risk in the IoT environment is dependence. With the evolutionary increase in the number of IoT devices, the communication among the devices become complex since they no longer communicate only by explicit pinging but implicitly as well by using services like IFTTT (If this, then that).
   In IFTTT, the company offers a software platform that connects the devices, applications, and services belonging to diverse developers to each other to initiate one or more automation involving those devices, applications, and services. For example, the automation happens like, if one makes a phone call on his/her android phone, then a call log will be added to Google Spreadsheet, if smoke is seen, then turn lights to red color, If I am out of home, and sight hound detects a person, turn lights to red color, If a thermometer senses the room temperature to be higher than the threshold and the smart plug detects that the AC to be switched off, then the windows would automatically open [3]. This feature is called interdependence or implicit dependence of the IoT devices.
2. **Heterogeneity:** The different kind of protocols used among the devices, range of interfaces and firmware employed, their varying storage capacities, the various access control mechanisms employed and the different authentication and communication protocols that are utilized make heterogeneity an important feature of IoT devices.
   This heterogeneity in the hardware, software, and process requirements is justified by the diverse functions of IoT devices. The protocols employed in IoT can range from being completely free to consortia-driven standards such as ZigBee or WirelessHART, to completely proprietary. Another reason for this heterogeneity is the wide variety of applications covered by IoT that require the different number of devices to operate; different communication ranges for their devices, different latencies and reliabilities, varied network coverage, and traffic loads [4]. Also, the applications might require utilizing diverse energy sources and having distinctive prerequisites on energy proficiency and lifetime [4].
3. **Constrained:** The IoT devices are designed to meet different requirements and as such, are diverse. For example, the implantable medical sensor devices have to be small in size as well as lightweight, implying that their computing abilities and storage capacities will be little. Similarly, the devices meant for defense purposes have to remain deployed in war zones, implying that their batteries cannot

be changed now and then and that their power consumption has to be less [2,3]. The same limitation applies to devices installed in the agricultural and industrial fields. Also, the devices utilized in the genre of robot control systems and automotive vehicle systems need to work under strict deadlines and hence are constrained by time. In essence, it can be said that constrained is one of the basic features of IoT.

4. **Pervasive:** It is estimated by Cisco that by 2020, every person would be surrounded by an average of 6.58 devices [2], which makes a humongous approximation of 50 billion IoT devices by 2020. As IoT devices would soon be seen everywhere, human beings would find themselves dependent on these devices just like air and water [3]. The feature of IoT to exist everywhere is referred to as Pervasiveness. Moreover, due to their rapid proliferation, the amount of data that IoT devices produce, send, and use go to the astronomical figures. Let us take the example of a supermarket where every item is Radio Frequency Identification (RFID) tagged. The raw RFID data format stands like EPC, Location and Time where EPC is the unique identification that is read by the RFID reader; location marks the place where the reader is positioned, and time represents the time when the reading was performed. To save any raw RFID record, 18 bytes of storage are required. Let us suppose; there are almost 700,000 tagged items present in the supermarket, hence if the supermarket possesses readers that scan the items every second, about 12.6 GB RFID data will be produced per second which makes to a whopping 544 TB in 24 hours [5]. Hence, for managing, analyzing, and mining RFID data, effective methods must be developed.
5. **Unattended:** Implantable medical sensors, sensors installed in the battle fields, the smart meters, the devices used in agricultural and industrial areas have to perform their functions for long periods after they are installed and because of the nature of their functions, they remain unattended during these periods [3].
6. **Affinity:** As the wearable devices and smart home products become commonplace, the privacy issues creep in; IoT devices not only collect the information such as pulse, blood pressure, etc. but also tend to record the environmental conditions like the places you have visited, the temperature of the room, etc. The sensors deployed on the roads to measure the levels of noise can record the conversation of 2 individuals and thus pose a threat to their privacy. Similarly, when people give consent to save their credentials to allow the smart TV to automatically download the content of your choice, a strong security and privacy breach can happen just by hacking onto that TV. This feature is thus named affinity since the IoT users and the devices share a close relationship with each other.
7. **Mobility:** Many IoT devices can roam from one place to another, e.g., wearable devices move as the individuals move. Similarly, the smart vehicles move from one district to another, collecting road information as they move.
   As per the International Telecommunication Union (ITU), the number of mobile users today in the world stands at a staggering figure of 7.3 billion. It is not possible to manage and support these devices using the old versions of the IP protocol. Hence newer versions like IPv6 over Low Power Personal Area Networks (6LoWPAN) were developed to support mobility and other constrained features of IoT devices [6].

3. How IoT differs from WSN. One of the major empowering technologies of IoT is the Wireless Sensor Network (WSN) [7]. The sensors used in WSN are curbed resource wise [8] as are the end nodes in IoT. Moreover, similar challenges to the design of a security system exist between WSN and IoT. Nonetheless, security issues in WSN are less challenging as compared to those of IoT [9], and thus, the security solutions applicable to WSN do not fit IoT. This is well explained by the exclusive differences in the targeted applications of the two and their distinctive characteristics as pointed out in Table 3.1.

- Primarily, the most famous and targeted applications of WSNs include the ones requiring data collection, e.g., surveillance [15] and environmental monitoring [16]. In these systems, the WSN sensors collect data and transmit it using the multi-hop routing protocols [10] to the WSN sinks. This communication is unidirectional; the reverse direction is used only to manage the sensors by sending them the control messages, i.e., the control messages only provide instructions for the sensors and do not control or modify the associated physical world; thereby WSN doesnt have a significant impact on it [9]. On

TABLE 3.1
Comparison of WSN and IoT Features.

Features	**WSN**	**IoT**
Impact on the physical world [9]	Insignificant just monitor the surroundings	Significant impact
Heterogeneity [9]	Made up of homogeneous devices	Communications as well as devices are heterogeneous
Communication [10]	Unidirectional	Bidirectional
Privacy Expectations[11]	Less	Very high
Scalability [11]	Large scale	Extremely large
Interdependence	Not present among applications	Applications highly interdependent
Mobility [11]	Node is said to be mobile only when it moves inside a sensor network	It is said to be mobile not only when it moves within the network but also when it moves between various service providers at the network layer
Things identification [12]	Not required as the main focus of WSN is the correlative acquisition of data	Unique identification is a must for establishing communication
Internet connection [11]	Not necessary, usually connected via a wireless connection medium	A mandatory feature.
Constraints [13]	energy	Computational, storage, and energy.
Closeness to the owner	The devices used in WSN do not share a close bond with the owner	A very close relationship with the owner is created
The Protocol used at physical/ perception layer for communication [14]	Wireless Fidelity (Wi-Fi)	6LoWPAN
The Protocol used at the network layer for communication [14]	Transmission Control Protocol (TCP)	Datagram Transport Layer Security (DTLS)
The Protocol used at application layer for communication [14]	HyperText Transfer Protocol (HTTP)	Constrained Application Protocol (CoAP)

the other hand, there is a strong coupling between the cyber world and the physical world in the IoT systems. As a result, IoT puts a considerably noteworthy impact on the physical world, and hence, it is necessary that the security of the physical system be considered as part of the security design.

- The sensors in WSNs, as well as the end nodes in IoT, are constrained of resources; while WSN sensors usually face constraints only in terms of the energy availabilities [13], the IoT devices suffer from a plethora of such constraints (memory, energy, computability, etc). As such, the device centric security mechanisms (software patches or anti-viruses) cannot be expected to be used in IoT. For example, the storage space required for a mere antivirus exceeds the total RAM of a normal IoT device, e.g., Common touch antivirus demand 128 MB RAM whereas most IoT devices own a single-threaded microcontroller (8051,MSP430, ATMEL series) that has got less than 2 MB RAM.
  This makes the security of IoT end devices more challenging. Since they cannot support encryption algorithms, frequency hopping communication [17], anti-viruses, etc. lightweight encryption is employed for IoT devices.
- The sensors in WSN are mostly homogenous [9], but there is a huge factor of heterogeneity involved in IoT as the devices vary in the type of protocols they use, architecture, size, functions, operating systems, etc. This makes it tricky to build a generic security solution for these heterogeneous IoT devices.
- Representing one of the peer-peer ad-hoc networks, WSNs are generally designed for one particular application and each WSN remains detached, and works independently of other WSNs [15]. On the other hand, the essence of IoT is that it interconnects multiple domain-specific autonomous systems, including WSNs.
- The scalability of IoT is huge, hence to maintain the key management system is difficult. The heterogeneity of devices makes the process even more complex in IoT. The most famous key management scheme used in WSN is the random key distribution [18, 19]. The scheme has enough scalability to support WSN but not good enough to support IoT scalability.
- Moreover, it utilizes a centralized key pool which lacks in IoT, thereby making it extremely difficult to

apply random key distribution in IoT. Another famous key distribution mechanism is the polynomial based key pre-distribution [20, 21] but it cannot be applied in IoT because it demands heavy computational overhead and a lot of memory as well. Therefore, new key distribution mechanisms are needed to be built for IoT.

- Finally, the data collected by WSN applications is less human-related as compared to the data collected by IoT applications. Therefore, on analysis of the data transmitted by the IoT devices, personal information of people can be deduced, making privacy a huge concern.

Hence, we can conclude that the Security and Privacy issues and requirements in IoT are much higher than those of WSNs and thus the security solutions meant for WSNs cannot be adapted in the environment of IoT.

4. IoT applications with their security requirements and challenges. In this section, we illustrate five important IoT applications identified by IERC and highlight the services, particular security requirements, and challenges of each application. This section brings into highlight the vast services offered by IoT but also hints towards the major security and privacy hurdle that comes along with this comfort and ease.

4.1. Smart City. While there remains a conflict on any single definition of a smart city [22, 23], its common contemporary understanding brings us to the following Smart City features: Smart Energy, Smart Mobility, Smart Healthcare, Smart Economy, Smart Homes, Smart Information Communication and Technology (ICT), Smart Infrastructure, Smart Citizen and Smart Governance. The central motivation for the building of smart cities is to raise the quality of living of its citizens. In simplest terms, the smart city can be described as a city planning approach that banks significantly on Information and Communication Technology (ICT) to monitor and subsequently integrate and optimize the conditions and usage of citys lifelines like roads, bridges, tunnels, railway lines, seaports, airports, electricity, water, communication , etc. and an approach that effectively plans their management.

By keeping an eye on all the major systems, better decisions could be expected from the colossal streams of enormous data. For example, when home appliances like refrigerators, and washing machines are controlled by IoT, better energy management is obtained. Also, when the trees, plants, air, and the environment get monitored in a non-obstructive manner, optimal quality work environment could be expected. Cities keep on attracting new people, and by 2030, the UN assesses that more than 60 percent of the worldwide populace is assumed to live in huge cities [24]. With almost 38 million individuals, Tokyo stands at the pinnacle of most crowded cities of the world taken after by Delhi, Shanghai, Mexico City, Sao Paulo, and Mumbai. The results and difficulties for such huge increment in populace on the city assets and administrations are more than self-evident. The only feasible solution is to stand up to this issue by creating strategies to lessen the asset utilization of the city in a savvy and clever way. Figure 4.1 illustrates some of the most important smart city services, along with the challenges that require addressing.

4.1.1. Security requirements and challenges. The security requirements claimed by the smart city are shown in figure 4.2 while the major challenges include:

1. Extreme Heterogeneity: In the form of huge number of different sensors deployed in the city which are brought together in a single smart city eco-system [2]. Developing a generic security procedure here is challenging.
2. Scalability: Since there is a multitude of available devices, attack surface is also huge.

Therefore, it is exposed to all the threats & challenges posed by these features, which are described in detail in section 5.

4.2. Smart Health. Today a significant rise in the proportions of aging populace is witnessed. As such, IoT Health monitoring Systems (HMS) have been developed to provide a feasible contrasting option for dealing with healthcare instead of traditional approaches. The intent of HMS is to provide cheap remote healthcare to people who need it, thus maintaining their independence as well as avoiding hectic and skyrocketing costly interactions with healthcare institutions. Apart from HMS, other services [25] of smart health include:

- Activity of Daily Living (ADL): Grooming activities like brushing teeth, face washing, making hair, eating, dressing, sleeping, toileting, etc.
- Instrumental Activity of Daily Living (IADL): preparation of meals, laundry, use of medicines, housekeeping, shopping and etc.

FIG. 4.1. *Smart City services and challenges.*

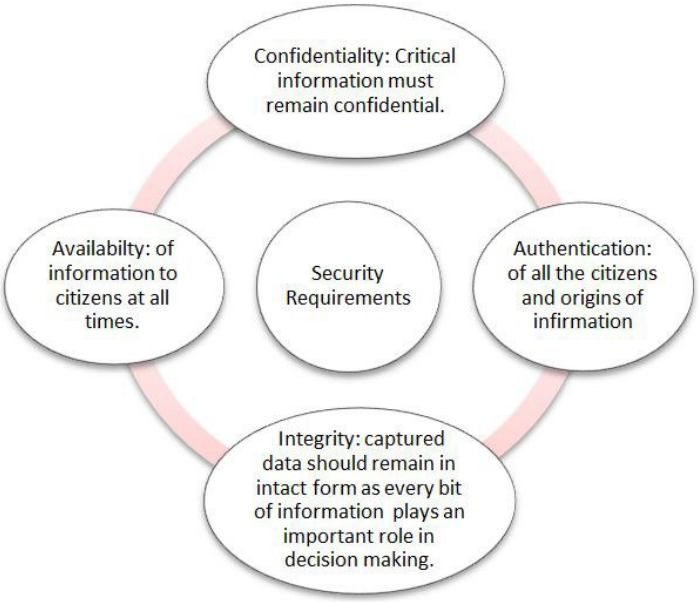


FIG. 4.2. *Smart City security requirements*

- Ambulatory Activity of Daily Living (AADL): Static activities like lying, standing and sitting, dynamic activities like walking, running, jogging, bike riding and etc., transitional activities like standing to sitting, sitting to standing, standing to walking, etc.
- Monitoring of mental functions (MF): Memory, judgment, understanding, sense of direction, etc.
- Physiological activities: monitoring of heart brain and muscle working.
- Social Activities of Daily Living (SADL): get together with family and friends, making phone calls, video calls, etc.

As per the United Nations Population Fund (UNFPA) [26] there would be more than 2 billion people all around the world who will be aged more than 60 by the year 2050. Also, World Health Organization (WHO) says that by 2035, the world would be 12.9 million short of healthcare personnel [27]. Age itself becomes a

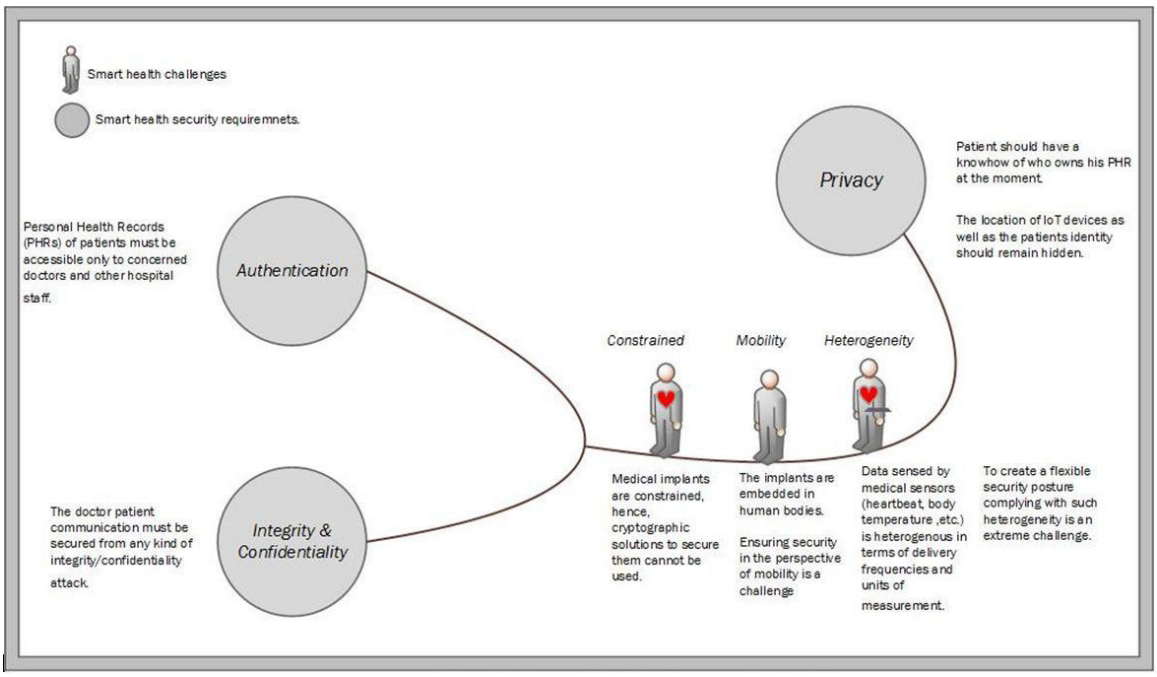


FIG. 4.3. *Security requirements and challenges of smart health.*

significant criterion of risk for developing chronic diseases like dementia, alziemers, diabetes, cardiac problems, osteoarthritis, etc. [25]. Also, the aged people may face an elevated danger of falling and sustaining hip fractures [28]. However, there are not enough resources available to deal with this type of sensitive care [29, 30]. As such, it is very important to have smart healthcare development. Nonetheless, smart health environment provides benefits; it also suffers from various challenges.

4.2.1. Security requirements and challenges. Given the medical implants usually remain unattended for long durations, the S&P requirements and challenges [31] in the light of IoT intrinsic features are summarized below in figure 4.3. The holistic impact of these challenges would be visualized in the next section when the threats/challenges raised by each of the feature are studied in detail.

4.3. Smart Building. One of the major building blocks of a smart city, a Smart Building is the one in which all the service systems are controlled automatically and are integrated with each other, working co-operatively in order to optimize the utilization of resources and boost the savings of vested money and operating costs, flexibility and performance [32, 33].

With the advancements in the technology, smart buildings were induced with the ability to self-learn, change, and adjust their performance as per the requirements of the environment, organization or an individual [34]. The vision of connecting various things to the internet is brought into practice with the use of various applications that offer remote monitoring and control of these devices. But regardless of the presence of smart buildings and smart technologies for quite some time now, their prevalence is not widespread and hence, their potential is not fully tapped. This is because there are still a lot of hurdles that exist in the way to the exact realization of smart buildings.

4.3.1. Security requirements and challenges. The vital security requirements of smart buildings / smart homes (SH) are sketched out diagrammatically in figure 4.4 while the critical security challenges include:

1. Heterogeneity: Bringing different technologies together can give rise to new security threats [35].
2. Context Awareness: If a thing moves to a new location or its environment/context changes, the SH system must be able to both detect as well as react to it. Making of such an adaptable security solution is challenging.

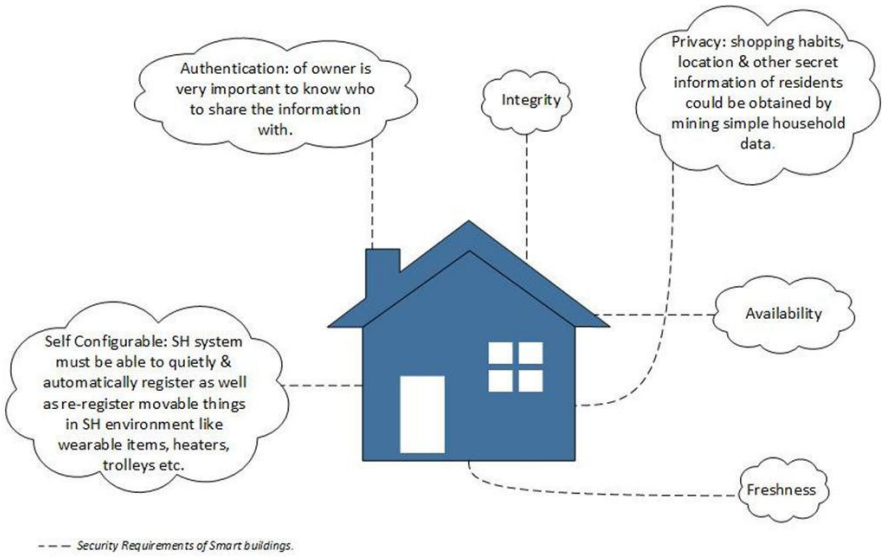


FIG. 4.4. *Vital security requirements of smart buildings*

3. Usability: The easy to use and easy to learn feature must be there in the SH system. Designing a simplistic security posture is challenging.
4. Internet Theft: Stealing private photos from clouds, video content from IP connected house cameras.

4.4. Smart Transport. The development of Intelligent Transport System (ITS) has paved the way to the creation of smart cars, smart bicycles, smart buses and trains [36] by equipping them with various types of sensors and actuators that include radars, cameras, Global Positioning System (GPS), Event Data Recorders (EDRs), omni-directional antennas, Electronic License Plates, Electronic Chassis Numbers, etc. [36].

With huge number of these autonomous and highly sophisticated vehicles hitting the roads, researchers are considering ways to smarten and tidy up the roads on which they travel. Smart cars on smart roads would offer advantages like notifying drivers about the empty parking slots via their mobile phones, inform cars about the road conditions, weather conditions, traffic awareness services, wildlife movement patterns, etc. The way to getting it going is an IoT system that incorporates sensors (wired/wireless) installed in the roadway and on existing traffic lights.

The vehicles are loaded with On-board Units (OBU) that communicate with other vehicles using Vehicle to Vehicle (V2V) communication and with Road Side Units (RSU) that are installed on the sides of the roads using Vehicle to Infrastructure (V2I) communication [36]. The applications in the transport industry incorporate the utilization of smart things to screen and report different parameters starting from pressure in tires to the distance from other vehicles. Radio Frequency Identification (RFID) has been utilized to aid in streamlining vehicle generation, amplify co-ordination among vehicles, upgrade quality control, and enhance client services [37].

The use of Dedicated Short Range Communication (DSRC) will conceivably help in avoiding interference with other devices as well as in accomplishing higher bit rates. V2V and V2I communications will essentially progress ITS applications, like vehicle safety applications and traffic management services, and will be completely integrated into the smart transport infrastructure [37]. Smart transport offers a lot of services to ensure efficiency and safety of travel but at the same time suffer from a lot of issues. Figure 4.5 explains the various aspects of smart transport.

4.4.1. Security requirements and challenges. As already stated above, to achieve various smart transport services, V2V and V2I communications are used, but to secure these communications, several security requirements need to be studied comprehensively [38]. Figure 4.6 summarizes the various security requirements and challenges that are faced in the realm of ITS.

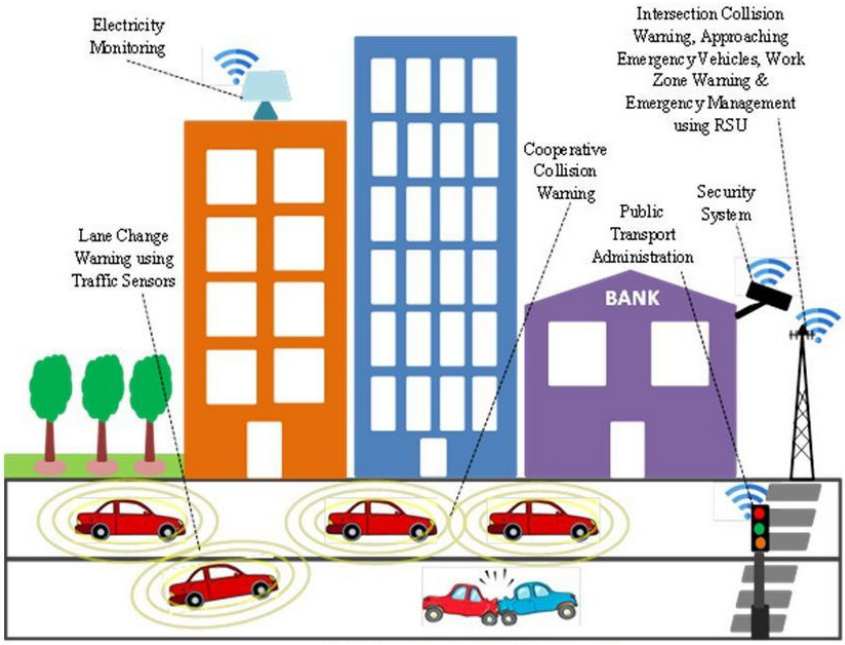


FIG. 4.5. *Smart transport services in a smart city*

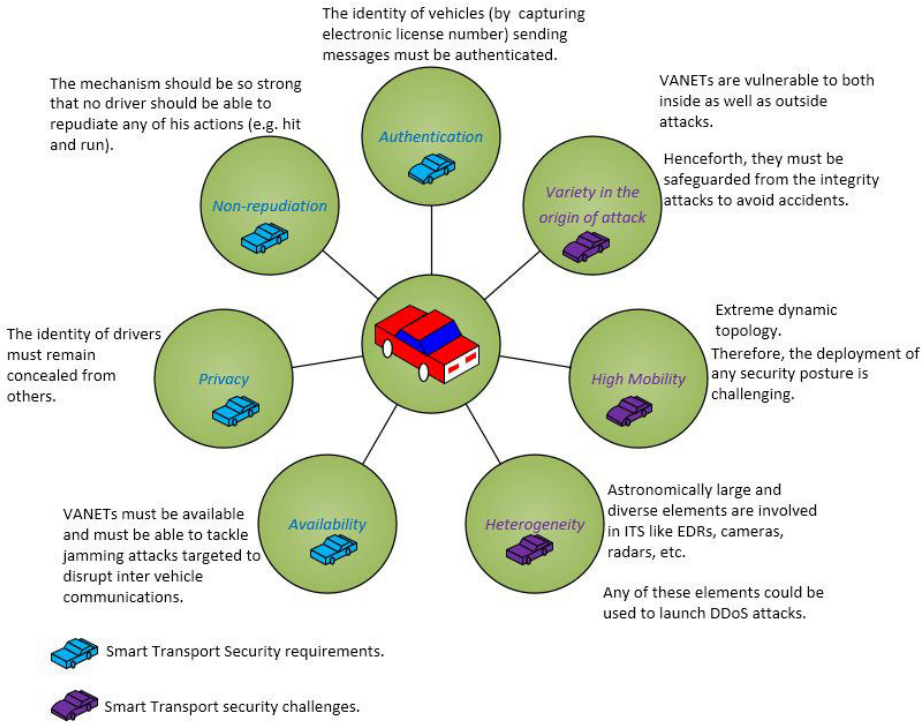


FIG. 4.6. *Smart transport security requirements and challenges*

4.5. Smart Industry. The manufacturing industry will soon witness a revolution as their mode of production will shift from digital to intelligent [39]. This is attributed to the fast improvement of electric and electronic innovations, the manufacturing technology and information technology [40]. To become the best players in the smart industry development, a lot of industrialized nations are profoundly giving careful attention to the clever manufacturing technology [41]. For example, China Manufacturing Plan [42], Industry 4.0 Strategy [43], Europe 2020 Strategy [44], USA Reindustrialization and Manufacturing reflow Strategy [45]. Some of the most important services of smart industry include water monitoring, transport assessment, manufacturing, retail, electricity monitoring, gas and oil monitoring, worker safety services and location services.

Although an impressive growth is witnessed in enhancing the flexibility, quality, and efficiency of the manufacturing systems, a huge risk in this race to achieve the smartness is that of security which is viewed as being an optional concern instead of a basic part of the procedure of development and deployment.

4.5.1. Security requirements and challenges. The industry systems are one of the most targeted victims of attackers [46]. The security requirements and challenges particular to smart Industry are highlighted in figure 4.7 and explained below:

1. Confidentiality: Industry data should be known only to its owners and must be hidden from others. Espionage attacks are a commonplace in Industrial IoT (IIoT) as other companies want to know what their contemporaries are up to. Hence, data, code, and system configurations must be secured.
2. Integrity: To prevent accidents in the industrial units, it is very important that the integrity of exchanged data must be maintained.
3. Availability of the system: Denial of Service (DOS) and Distributed Denial of Service (DDOS) attacks could be easily launched, but the industry manufacturing systems must always remain in the operational state.
4. Authentication: It is necessary that every part that is involved in the manufacturing process is authenticated.
5. Lack of standardization: No standard protocol exists among the industry systems in general and SCADA systems in particular. In fact, there are almost 150-200 open standards.
6. Cyber-physical attacks: Trojans, viruses, worms, Dos, DDOS user-compromise, and root compromise attacks could be launched easily. Dos attacks compromise the sensors and stop them from sending any data. Such attacks could be launched by either disrupting the communication channel or the routing protocol
7. Scalability: As the number of industry units increase, the probability of attack also increases because the attack space increases.

Next section illustrates how the security requirements and challenges actually creep into the various implementations of IoT. The very features of the devices used in these applications make them insecure.

5. Practical illustrations of Security and Privacy breaches in view of IoT Features. In this section, the most important as well as famous attacks, threats and challenges are studied in the light of IoT features, i.e., the threats they cause, and the challenges that exist because of their impact. All these features are exhibited by the devices used in the implementations discussed in section 4. Some of the existing solutions from literature are discussed and their critical analysis is presented in this section.

5.1. Threats, Challenges and Solutions in Interdependence (Implicit Dependence).

1. **Threats caused by Implicit Dependence:** There are potentially three inherent security issues linked to the use of IFTTT [47]. a) No consideration to security-related context of IoT end devices b) causing ambiguity by assuming that different applications/services work independently and c) vulnerabilities that arise because of the incomplete specifications provided by the user because of their incompetence to understand the cross-device relationships and their effects. Henceforth, even if the attackers actual target has a strong defense mechanism, it can be compromised because of this feature.
   E.g., in case of smart buildings/homes, both the smoke detector and sight hound could be active at the same time creating ambiguity for people to decipher whether the lights turned red because of smoke or because of an intruder. Also, in case of the opening window scenario discussed in section 2(a) the hacker neither requires to handle the automatic window control nor the thermometer. S/he just needs

FIG. 4.7. *Smart industry security requirements and challenges*

to compromise the smart plug connected to the public network and make it switch off the AC. The temperature of the room will automatically increase, and consequently, the windows would open - a dangerous physical security breach [3]. Also, different interdependencies and contexts demand different levels of S&P.
In the case of smart transport, the dependence of vehicles on the information coming from RSUs could become dangerous. If an RSU is hacked for instance and there is a blind corner. A car is speeding in the wrong direction. Now, instead of asking the vehicle coming from the other side to slow down/stop, the hacked RSU tells it to go as it pleases as there are no cars on the other side. The result will be a fatal collision.

2. **Challenges:** The researchers usually try to protect single devices rather than creating a clear defensive boundary for them. This results in an adverse effect on the security aspect of IoT. It is, however, difficult to define a defensive boundary for them because of their interdependence, which makes it difficult to set a clear set of permission rules for these devices.
ObjectivesTo study the anomalous behavior of cross-device interdependence in IoT.
Device new security policies for differentiating normal behavior from anomalous.

3. **Solutions** The traditional security approaches like anti viruses, software patches would be inefficient in the IoT world because of the implicit dependencies shared by them. Table 5.1 depicts the various IoT solutions given in diverse fields to deal with issues caused by the interdependence feature of IoT. From the problems, it is concluded that more practical and effective solutions are the need of the hour.

5.2. Threats, Challenges and Solutions in Heterogeneity.

1. **Threats:** IoT security report, 2015 [50] indicates that >90% of IoT devices have hard-coded key vulnerabilities, 94% have web security vulnerabilities in their web interfaces implying that they can be easily attacked by the hackers. Moreover, the protocols used in IoT do not have tough security procedures implying the protocol vulnerabilities could be exploited easily [51], and since these protocols greatly vary in their semantics, when they work together erroneously, other security threats could arise like Bad Tunnel [52]. The Bad Tunnel attack is launched by persuading the victim to open a URI using a Microsoft edge web browser or internet explorer or to open an office document. Once the victim does so, the attacker can camouflage like a file server or a local printer, circumvent the explorers sandbox or take the download update of windows, network traffic, and certificate revocation lists under its control and could be launched on all the versions of internet explorer and Microsoft office. Table 5.2 describes

TABLE 5.1
Critical Analysis of Solutions to Interdependence Issue

References		
Tianlong et al. [47]	**Domain Studied**	Smart Home
	Interdependence	If there is a fire alarm, open the windows. The rogue player can try to compromise the fire alarm to break into the house.
	Advantages	Provides a fresh roadmap to look at the security disaster of IoT in a new light: thinking beyond the traditional approaches of security.
	Problems with the solution	A security posture is defined for every device separately for detecting whether the device is acting normally or not. The solution becomes absolutely impractical and complex when the number of devices increases, hence not suitable for IoT.
Yunhan et al. [48]	**Domain Studied**	Samsung Smart Things platform.
	Interdependence	A strong defense mechanism might be present in a smart phone, but when the apps are installed on these phones, people tend to give various permissions to these apps; the interdependence (over-privileged problem) is then exploited by the hackers to break in and cause damage.
	Objectives	To provide a permission system based on context to alleviate the over-privileged problem in appifiedIoT environment. Fine-grained control of application behavior is achieved. Provide the user with important information such as run-time data, procedure control, and data flow of each IoT device and then allow the user to either allow/reject the action.
	Advantages	Provides contextual integrity. Is backward compatible and hence can be easily taken up by the present IoT platforms. Flaws like thefts and break-ins in permission systems like smart phones have been identified. Misbehavior by the attackers will be detected very early.
	Problems with the solution	The Final decision is made by the user. So, if he makes a wrong decision and says allow where he should have said deny, the choice is remembered by the system, and the user is not prompted the next time such an attack occurs. Hence impractical in IoT.
Luca et al.[49]	**Domain Studied**	IoT Health.
	Interdependence	If a person falls, then the relatives, nurses, and other medical staff would be informed. A fall like situation can arise when a person drops himself on a sofa or lies on a bed (tries to deal with the ambiguity problem).
	Objectives	Create an alarm system to deal with sudden ailments and falls of elderly people. Aims to provide the perfect position of individuals (indoor & outdoor), monitor their vitals and activities.
	Advantages	An Omission of costly hospital charges for the care of the elderly by the use of wearable technologies.
	Problems with the solution	Consideration to S&P is completely left out. The problems caused by the interdependence could be immensely exploited. Via a huge attack space that is available in IoT, an attacker can easily attack the wearable device and create false alarms and false notifications to take medicine in huge quantities, thereby can lead to fatal outcomes. The wearable devices can pose privacy threats as well. A person could be tracked down to his exact location, which could lead to a privacy breach, and at the same time very dangerous.

various types of web security vulnerabilities and what percent of these traditional vulnerabilities still exist in IoT.

2. **Challenges:** Because of this heterogeneity among the IoT devices, a single defense posture wont suffice in the IoT environment. Researchers need to dig out the general security mechanisms somehow.
3. **Solutions:** The solutions should offer a way to manage the variety of devices/technologies/services /environments to tackle the possible vulnerabilities of diverse IoT devices. Table 5.3 highlights some of the proposed solutions, their advantages, and loopholes.

The proposed solutions become practically unsuitable for large scale analysis and have some other flaws which make them incomplete. Also, most of the research is focused on using classical Intrusion Detection

TABLE 5.2
Web Security Vulnerabilities

Web Security Vulnerability	**Description and Effects**	**Vulnerability %age in IoT**
Cross Site Scripting	Malevolent scripts are infused into generally favorable and confided websites.	55.5% [50]
File Manipulation	The contents of a file are modified in a way to cause the application to start erroneous processing thereby displaying horrible results like throwing the application in an unstable state, disclosing confidential information, overwriting the file, etc. [53].	12.5% [50]
File Disclosure	The Attacker tries to hack down the entire path of the file and disclose its contents. This can be done by eating the cookies or making the web application do something that is not intended [54].	4.6% [50]
File Inclusion	Application fabricates a way to executable code utilizing an aggressor controlled variable in a way that enables the attacker to control which record/file to execute at run time.	5.7%[50]
SQL Injection	SQL statements are infused with malicious lines of code. Executed through a web page input, SQL injection can destroy a complete database.	4.9%[50]
HTTP Response Splitting	Refers to the state when an application is not able to decontaminate the input values. Leads to various other vulnerabilities like cross-site scripting.	1.4%[50]
Command Injection	If the application is vulnerable, it can be exploited to run arbitrary commands as it allows the mischievous cookies, HTTP headers, etc. to pass into the system shell, thereby giving the attacker rights that it dreamt of having. If the attacker is able to insert a single delimiter like ; that marks the end of a command, it can insert its command and get it executed [55].	10.4%[50]
Code Injection	Is different from command injection in a way that the attacker needs to insert his/her code, which is then executed by the running application. Achievable by the poor data validation approach of applications. It can lead to the loss of integrity, accountability, confidentiality, and availability [56].	1.9%[50]
Possible Flow Control	Change the order in which statements execute. Usually done by altering the program counter.	1.6%[50]
Unserialize()	Unserialize is a function that takes a single serialized parameter and transforms it into a PHP value. If the suspicious input is passed to the unserialize, it can result in object instantiation and auto loading, allowing the attacker to exploit it as he wishes [57].	1.3%[50]

Systems (IDS) and Intrusion Prevention Systems (IPS) for protecting a diverse range of devices all at the same time. However, heterogeneity of the IoT devices doesnt let it work because the attacks may vary in their character depending on the device they target.

More suitable IDS and IPS systems for IoT devices which exhibit heterogeneity need to be studied further. It is concluded that full-fledged solutions to the problem posed by heterogeneity are currently absent, and more work needs to be done in this direction.

5.3. Threats, Challenges and Solutions in Constrained.

1. **Threats:** Since IoT devices are mostly constrained by resources, storage capacity, battery back-up, and time delay, they are unable to support the necessary defense of the system as well as the network. Memory Management Unit (MMU) is absent in the lightweight IoT devices, hence the functions such as memory isolation, Address Space Layout Randomization (ASLR) and other types of memory safety procedures cant be installed on these IoT devices [3]. Also, the existing encryption and authentication algorithms are heavy weight requiring huge computing resources. If the devices start utilizing their computational and other resources on performing these heavy weight operations, then they would be left with very little energy and resources to perform their intended operations.
   As a result, an easy attack space is offered to the mischief makers for compromising these IoT devices. In fact, many IoT devices communicate with the server without checking its certificate and without any encryption only to save their resources. A man-in-the-middle attack can be launched with ease apart from the interception of communication happening between the two parties.
2. **Challenges:** Designing of lightweight security solutions for constrained devices is a challenge.
3. **Solutions and opportunities:**
   - The author in [66] proposed a lightweight software fault isolation procedure on tiny embedded

TABLE 5.3
Critical Analysis of Solutions to Heterogeneity Issue

References		
Costin et al. [58]	**Type of Heterogeneity**	Firmware of 32000 distinct devices.
	Objectives	To perform a static analysis of 32000 firmware images.
	Advantages	Found 38 formerly unknown security vulnerabilities. Found that these vulnerabilities were affecting almost 1,40,000 devices on the internet.
	Problem with the Solution	Suffers from the problems of static analysis: production of false positives because of generic analysis and false negatives because of much specific analysis packed or obfuscated code, etc. A particular programming language domain is targeted like PHP, C, etc. In reality IoT equipment can contain a mixture of programs written in various languages.
Drew et al. [59]	**Type of Heterogeneity**	Different firmware of IoT devices.
	Objectives	To provide a dynamic, complete and sound analysis of firmware programs. To find potential bugs, and security threats.
	Advantages	Symbolic execution of firmware programs to investigate their security. Exposed 20 memory safety bugs and one peripheral misuse bug.
	Problem with the Solution	The Complete analysis can be unmanageable in many firmware programs. The techniques employed: state pruning and memory smudging are not enough. Imprecision factors exist among the sources: disparity in the execution of firmware (natively or symbolic execution), false positives and false negatives can arise when the proposed system executes states that are actually never reached in reality.
Zaddach et al. [60]	**Type of Heterogeneity**	Different firmware of IoT devices
	Objectives	To perform an elaborate dynamic study of firmware in embedded systems. To evaluate the performance on three real-world security scenarios: vulnerability discovery, hardcoded backdoor detection, and reverse engineering.
	Advantages	Performs dynamic analysis of firmware by giving direct memory access to the real device employing an emulator. Firmware relying on absolutely amorphous peripherals could be studied.
	Problem with the Solution	Unable to imitate all real device actions, it makes use of emulators. The need to have an emulator device for any device that is under test puts a heavy fiscal burden. Incurs serious hurdles for large scale analysis.
Daming et al. [61]	**Type of Heterogeneity**	Variety of device firmware.
	Objectives	To assess FIRMADYNE across a huge dataset of 23,035 firmware images for exposing security vulnerabilities. Aimed at the Linux operating system.
	Advantages	Enable large scale and dynamic firmware analysis. No emulator required. If the vulnerability is detected, results regarding the necessary actions to be taken are provided. Automatic vulnerability verification is performed.
	Problem with the Solution	Works only on the LINUX based systems.
Virginia et al. [62]	**Type of Heterogeneity**	Different Access Control (AC) mechanisms.
	Objectives	To replace the Access Control List (ACL) based AC mechanisms by role-based AC mechanism for the reason that former AC procedures are hard to administer when the count of devices and resources increase.
	Advantages	Adds a median layer that functions in assigning privileges to roles and roles to subjects. In this way, the rights do not descend directly to the subjects but come through roles. The effort to manage AC lists is drastically reduced.
	Problem with the Solution	It requires a hurricane effort when the numbers of roles or resources grow. Impractical when a lot of domains are covered by AC systems.
OASIS [63]	**Type of Heterogeneity**	Different access control mechanisms
	Objectives	Specifies policies of Attribute based Access Control (ABAC). To support varied data types, name types, path expressions, and objectives for attributes (String, integer, internet-based name, etc.) Provides a system of modularization to accord with complicated policies.
	Advantages	Uses attributes of the subject like its location, age, position as well as its environment and resource properties to define access policies thereby eventually cutting on the cost of complex rules, the number of rules and rule changes. Slashes the processing and data availability needs.

	Problem with the Solution	Complex to manage. Increased probability of having defect because of heavy expressiveness of ABAC. Needs a persistent description of subject attributes both within a particular domain as well as across multiple domains. It is not always possible to have a well-defined environment, resource, and attribute description of subjects in the IoT domain.
Sergio et al.[64]	**Type of Heterogeneity**	Different access control mechanisms.
	Objectives	To develop an AC system that supports scaling and changing environment needs of the IoT.
	Advantages	An easy to use, scalable, feasible, and legible AC mechanism is developed.
	Problem with the Solution	There is delegation support in which a subject can award another subject with access rights, and provide it the right to grant further subjects. If one subject gets compromised or is corrupt, malicious subjects get the access rights and the entire systems come under a heavy security threat. X.509 certificates are used. Their management and encryption needs complicate the process and make it complex. The RSA encryption scheme is utilized hence impractical for IoT devices that are constrained of both space and processing capabilities.
Ki-Wook et al. [65]	**Type of Heterogeneity**	Different authentication and key management procedures.
	Objectives	Aims to provide a new Authentication and Key Management (AKM) mechanism for constrained IoT devices based on IEEE 802.11 key management and IEEE 802.1X authentication procedures.
	Advantages	No need for pre-configured security information between the IoT service domain and access network domain. Reduces the burden of constrained IoT devices for performing AKM by offloading the process to a strong agent. Computation and network cost reduced. Less memory usage.
	Problem with the Solution	Mutual authentication is performed only once. Only session keys are exchanged later between the stations and Access points (AP). This can be dangerous in the situation when an authentication users device gets hacked or becomes corrupt.

processors. The disadvantage of [66] however is that the performance overhead for the devices needing multiple address checking searches is huge and hence is not applicable for IoT devices expecting performance in real time.

- The author in [67] presents a complete, trusted computing functionality on low-cost embedded systems. However, its implementation requires the changes to be made in the existing hardware of the Microcontroller unit, so it cant be used directly on existing devices. Hence, novel lightweight algorithms need to be designed.
- New lightweight encryption algorithms are presented in [68-70], and modification in the existing cryptographic algorithm is presented in [71]. Yet, achieving the security of the same level by the lightweight algorithms is different and prone to new security issues.
- The Cloud computing comes to rescue [72] for dealing with the above problems as it allows centralized, shared, and scalable computing resources to be used on demand by any individual or organization. The amalgam of IoT and cloud can give strong processing power, huge storage capacity, and resource allocation in a scalable manner and on the fly deployment of applications with insignificant cost [73]. But, regardless of the advantages offered by this mix of IoT and cloud, it cannot be said that the cloud is the panacea of all the IoT issues. Firstly, the resources are centralized, implying the presence of a huge distance between the IoT devices and the cloud resulting in latency and jitter [74].

  Also, the physical distance gives rise to the inability of the cloud to access the local context based information like, the state of a local network, mobility pattern of the user, location information of a user, etc. Besides, because of this communication delay, real-time time-constrained applications cannot be accessed by the end users. Hence, there must be a new technological posture to extend the IoT to support time-constrained, location-aware, and mobility supported applications.
- The fog computing paradigm offers a way to cover up the gap among IoT devices and remote data centers by offering a distributed computing environment pushing the cloud to the edge devices of the network and thus provide benefits like efficient networking, easy data access, better com-

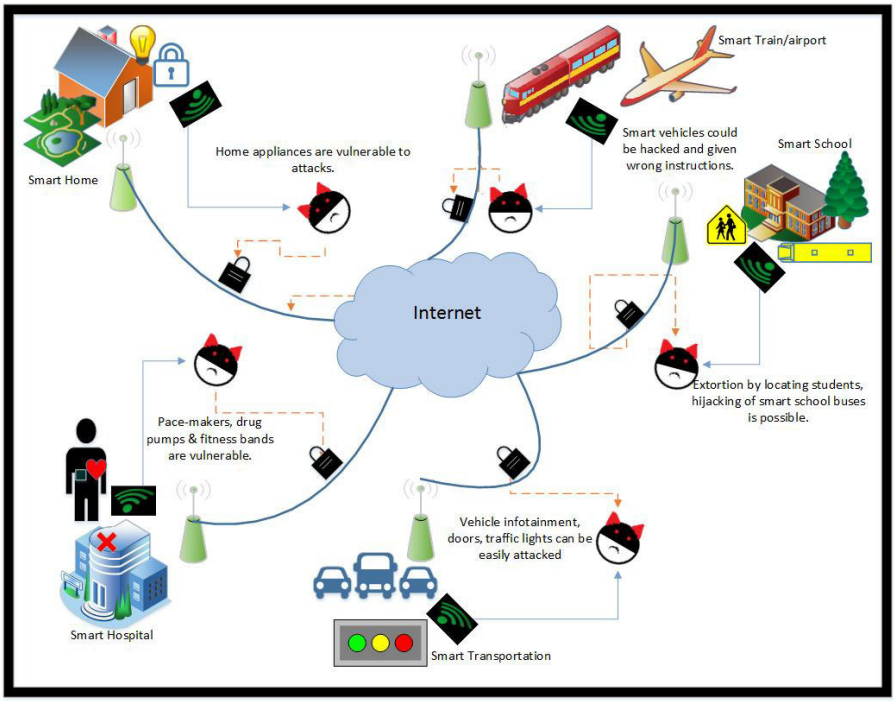


FIG. 5.1. *Examples of IoT botnets*

putation, reduced delay, storage, supporting heterogeneity, scalability, geo-distribution, locality, etc. [75]. However, fog computing suffers from the disadvantage of limited processing and storage capabilities.

5.4. Threats, Challenges and Solutions in Pervasiveness.

1. **Threats:** The MIRAI botnet of the year 2017 involved more than a million IoT devices where the attack traffic surpassed 1 Tbps. The botnet compromises the less secure IoT devices to achieve its goal of DDOS attack rather than computers. It was found by [2] that to launch huge scale DDOS attacks, IoT devices were employed. DDoS-for-hire services have lowered the barriers of entry for criminals to carry out these attacks, in terms of both technical ability and cost [76].
   As IoT penetrates into all the walks of life, i.e., industrial, agricultural, medical, etc. the IoT botnet target would no longer remain the website only, but will shift toward important national infrastructures, thereby causing grave dangers as shown in figure 5.1. Table 5.4 lists some of the botnet attacks that were launched in the recent past utilizing the multitude of available IoT devices. Also, the insecure configuration of these devices is a considerable threat to deal with.
2. **Challenges:** Since proper defense mechanisms are absent among the IoT devices, even the installation of an anti-virus is a hard task for them. Therefore, it is tough to detect and prevent IoT botnet in the early stages and thus a challenge.
   With the prediction of 50 billion devices by 2010, in addition to the scalability issues, the achievement of improvisation and optimization of IoT services on the internet would both remain a necessity as well as a hurricane challenge in front of the IoT professionals [91]. Moreover, in addition to the specific focus that cloud and fog computing paradigms would demand to increase the network efficiency and capacity, resource management will continue to remain a challenge.
3. **Solutions and opportunities:** The author in [92] distinguishes the legitimate user from the attacker based on the type of request that they send. According to [92] a legitimate user may send a request at low frequency and a proper content while as an attacker may send requests at high frequencies and with the same repeating content in all the packets. Yet, the assumption is really basic as the attacker

TABLE 5.4
Recent botnet attacks compromising IoT pervasiveness

Botnet Attacks		
Mirai botnet	**Launch date**	First seen on September 19, 2016, while the massive attack was launched against Dyn on Oct 12, 2016.[77]
	Description	Took advantage of over 600,000 less secure devices like web cameras, routers, baby monitors, etc. to launch a massive Distributed Denial of Service (DDOS) attack with attack traffic of 1 Tbps- largest on public record till Oct 2016. [77, 78]. Scanned huge blocks of the internet to find open telnet ports to hack into the devices using brute force methods of trying default username/password combinations which are seldom changed.
	Attack purpose	Launched by a Rutgers undergraduate student named Paras Jha, Mirai was one amongst the series of botnet attacks launched by him and his friends to make a profit out of DDOS attacks [78].
	Affected companies/ countries	Largest European hosting Provider Company named OVH. Dyn: A company providing Domain Name Services (DNS). Mirai launched on it brought down websites like Pinterest, Twitter, Reddit, Spotify, Github, affected Paypal, New York Times, BBC, etc. Krebs on security: independent journalists website who specializes in cybercrime.
	Monetary loss and other effects	$110 million in potential revenue was lost [79]. 8% of Dyn customer base chose to change their DNS provider after the incident [80]. HTTP flood and various other network-level attacks could be launched by Mirai botnet. Once the device gets infected by Mirai, it tends to remove any other malware on that device to claim the gadgets authority.
Reaper botnet or IoTroop	**Launch date**	On October 19, 2017, an Israeli security firm named Checkpoint announced about this new IoT botnet [81].
	Description	Built on top of the Mirai code with one significant difference, i.e., while Mirai used simple brute force method, reaper made a step ahead in the complexity of these attacks [82]. Utilizes actual software hacking techniques to find security flaws in the code of vulnerable devices to compromise them, i.e., while Mirai was looking for open doors to break in, reaper breaks open the locks on those doors [82]. It covers nine different known security vulnerabilities [83], e.g., by exploiting the CVE-2017-8225 vulnerability; the reaper gets access to devices .ini files where the important credentials are stored. This vulnerability is found in insecure cameras. Also, it spreads the infection to other devices like a worm. When the vulnerability is targeted, the device can be taken under the botnets control without raising any alarm.
	Attack purpose	AS per Arbor, the reaper is intended for use as a stressor service essentially catering the intra-China DDOS-for-hire market [84].
	Affected companies/ countries	Targets vendors like LinkSys, Ubiquity, Synology, Netgear, GoAhead, Avtech, D-Link, Mikrotik, and Vacron, among others [83]. So far, IoTroop/reaper has infected over 2 million devices across more than 1 million organizations.
	Monetary loss and other effects	Built on Lua engine and mixed with further Lua scripts (the embedded programming language that allows running of scripts), reaper code can be easily modified and updated to launch more attacks with more options [83]. When combined with some basic machine learning and AI techniques, future version of this malware would have the capacity to recognize basically any device it is confronted with, look for a related vulnerability in it and after that select a proper exploit for it and even have the capacity to build up a custom exploit [83]. With the emergence and entry of technologies like Swarm Intelligence into botnet configuration, Hivenets in which numerous compromised devices team up to work like one intelligent unit will be made [83].
Hajime botnet	**Launch date**	Identified by Rapidity networks in October 2016[85]. The infections have primarily traveled from Vietnam (greater than 20%), Taiwan (approximately 13%) and Brazil (almost 9%). So far, no high profile attacks have been launched by Hajime botnet [86].
	Description	Hajime is an IoT malware whose most important feature is that it blocks other botnets and has amassed an army of 300,000 compromised devices. [85]. It is hard to impede the Hajime operation because of its peer to peer and hidden botnet operation rather than a centralized one. Hajime has no attack code but only a propagation module. Currently in the benign state. Like Mirai, it brute forces its way into open telnet ports on various devices to compromise them.
	Attack purpose	While the botnet is ballooning up in size, its real purpose remains unknown [85].

		Table 5.4 (contd.)
	Affected companies/ countries	MikroTik [87].
	Monetary loss and other effects	The compromised devices could be used to launch different types of attacks on various websites ranging from DDOS to executing SQL injection exploits. The worm is currently in a no harm state but can block access to 23,7547,5555 and 5358 ports that serve as common entry ports for botnets like Mirai. In April 2018, Hajime was found to extensively scan 8291 ports in the bid to find devices running vulnerable MikroTik router OS and was even trying the Chimay Red HTTP exploit [86]. If it gets through this operation, it will install a new copy of itself on the victim node.
Ransomware Attacks RDOS (Ransom DDOS,e.g., WannaCry	**Launch date**	May 2017.
	Description	In any RDOS attack, the attackers communicate something specific to the owners threatening about the DDOS assaults on their organizational operations or contamination of the operational frameworks with Ransomware except if a particular ransom is paid by a specific due date. The ransom usually ranges from 5-200 bitcoins [88]. The threat messages are often escorted by brief attacks to let the victim organization have a glance at the attackers power [88]. Almost 86 countries faced ransomware attacks from April-June 2017[89]. One RDOS attack in China lasted for more than 11 days, and almost 47.42% of RDOS attacks were targeted towards China [88].
	Attack purpose	Motivated by financial gains, attackers here use the trick of extortion for making money.
	Affected companies/ countries	Countries: China(47.42%), South Korea, USA, Hong Kong, UK, Russia, Italy, Netherland, Canada, France [88]. Companies: Al Jazeera, Le Monde, Figaro, Bitfinex (Largest Bitcoin exchange).
	Monetary loss and other effects	Global financial losses from WannaCry reached $4 billion. Companies lose their customer base.
Persirai	**Launch date**	Discovered by Trend Micro in early April 2017 [89].
	Description	A security threat exploiting the vulnerabilities in computers through TCP port 81 and which has compromised almost 120,000 IP based cameras so far. IP cameras become the easiest targets for attacks because they use the universal plug and play protocol (UPnP), which allows them to open up a port and work like a server [89]. Once compromised, the attacker directs the camera to download malicious shell scripts from various sites. After that, Persirai attacks itself, deletes the installation files to hide its presence and runs only in the memory. The compromised camera on receiving the commands from the server automatically attacks other cameras utilizing zero-day vulnerability [89].
	Attack purpose	After gaining control of the cameras, the criminal can launch a DDOS attack on other computers using the User Datagram Protocol (UDP) floods. The attacker will provide the ports IP address where it wants to launch the DDOS attack and therefore can target any IP in the world.
	Affected companies/ countries	Out of 120,000 IP cameras that are compromised, 30% are from China, 3% from Italy, 3% from UK, 8% from USA [90]. 64.85% of cameras in Japan have been identified to be infected with a backdoor.
	Monetary loss and other effects	The Worst feature of Persirai is that the computers from which the command and control for running the malicious bots is executed use the country code of Iran ,i.e., IR. However, this doesnt indicate that the attacker is Iranian [90]. Organizations dont know that their cameras are utilized to launch the DDOS attacks.

may not always send the requests containing the same old content but may most probably simulate users requests with proper and different contents.

Besides, the IDS that are employed in IoT are actually meant for the traditional networks and hence dont work well in the constrained IoT environment. There is a dire need to develop IoT specific IDSs that are designed keeping in view the heterogeneity and the constrained nature of the IoT environment.

5.5. Threats, Challenges and Solutions in Unattended.

1. **Threats and challenges:** It is considerably impossible to monitor the state of these devices via an external interface given the condition in which they are deployed. Also, the functions that these unattended devices perform are crucial and tempting the potential attackers to attack them. An attacker can re-program a camera, for instance, to send the recorded data to it as well in addition to the actual server.

TABLE 5.5
Challenges posed because of the mobility of IoT devices

Challenge	Description	Cause	Effect
Increased mobility Signalling Cost [6]	Signalling cost refers to the cost incurred in managing the Tsunami of signalling traffic generated by the billions of mobile devices which operate today [96].	The devices may be sending the signalling data on a periodic basis. When that data is multiplied by the no. of mobile devices, the Signalling traffic reaches staggering heights [96].	Inefficient usage of resources. Extra stress is put on the network. Diminished Quality of Service (QoS) [96].
Packet Loss [6]	Refers to the loss of the packet before it reaches the destination. It can be calculated by using the formula: Packet loss= No. of packets lost/ total number of packets [97].	Network congestion. Weak radio signals. Corrupted Hardware. Cyber-attacks, e.g., Black hole attack and other DDOS attacks [97].	Decreased QoS Less Throughput Increased Delay because of the time spent in the retransmission of packets [97].
Handover Latency [6]	When a node changes its point of attachment from one network to another, it is called handover. The time spent in doing so is called the handover latency. The handover latency can be calculated by using the formula: Handover Latency = The last packet received from the previous point of attachment/first packet received from the current point of attachment[98].	Channel Detection. Authentication. Process movement. Duplicate Address Detection (DAD). Registration Association. Channel Scanning [98].	While the handover is being performed, additional delay in performing the mechanism can occur. Active connection to the network is disrupted because of these handoffs [98].
Greater End-End Delay [6]	The time spent from the point the packet was put on the channel by the source to the time it reaches the destination is called the end-to-end delay. A Very crucial issue for the applications requiring fast response [99,100].	Congestion in the network. Cyber-attacks.	Less QOS. Fatal consequences in case of hard real time systems.
Inefficient Energy Consumption [6]	Lessening the consumption of energy in constrained IoT devices is one of the critical challenges faced by the IoT community. [101]	Devices already have less energy. If their energy is spent on sending signalling messages periodically and in retransmitting the packets, the constrained devices would be left with very little energy to perform their intended jobs efficiently.	Device shuts down and doesnt perform the function it is expected to do.

2. **Solutions and opportunities:**
   - The author in [93] designed a system called Trust Shadow to make sure that a trusted environment is made available for the devices to execute their security-critical applications. However, it is based on ARM TrustZone technology using ARM-Cortex-A processors and hence doesnt support small IoT devices that employ lightweight processors.
   - Also, author in [94] proposes a mechanism for remote attestation of devices, however, it involves far greater delay affecting the normal execution of devices.

5.6. Threats, Challenges and Solutions in Affinity.

1. **Threats:** This feature poses a lot of security threats e.g. [95] indicates how an attacker can infer with confidence if the smart home is currently occupied just by mining the CO2 and smoke sensor data.
2. **Challenges:** To enjoy the services offered by various service providers one needs to share his/her personal information with the company, e.g., to give you the discounts, a vehicle insurance company wants to collect your driving data. Driven by profit, these companies usually store critical personal

TABLE 5.6
Various Mobility Management Protocols

Mobile IPv4 [103]	**Description**	Allows the node to adjust its point of attachment as per its need without having to alter its IP address. Foreign agents are required, i.e., an external agent that performs the mobility function on a nodes behalf in a foreign network.		
	Packet Reordering	No	**Mobility Scope & Management Class**	Global & Host-Based.
	Mobility Issue addressed	None [6]		
	Other problems with the protocol	Suffers from the fragility problem, i.e., it gets broken when the node has a single home agent (an entity that performs mobility and forwarding functions on behalf of a node in the network to which the node attaches itself in the) beginning and doesnt solve any mobility issue.		
Mobile IPv6 [104-106]	**Description**	An Enhanced version of mobile IPv4 with an extra-large address space. Doesnt require any foreign agent. Employs the use of binding cache mechanism to link a mobile nodes home address, i.e., the permanent address of a node present within the home network with its relative care-of address, i.e., the nodes new address in a foreign network.		
	Packet Reordering	Yes	**Mobility Scope & Management Class**	Global & Host Based.
	Mobility Issue addressed	Briefly addresses the issues of packet loss, handover latency and end-end delay. [6]		
	Other problems with the protocol	Doesnt solve the issues of high signaling cost and energy consumption. Hidden Terminal problems. No way to find the reasons for packet loss. Includes links that could be utilized only partly.		
Hierarchical Mobility IPv6 (HMIPv6) [107]	**Description**	Pressure on the speed of handover is witnessed in mobile IPv6 because of the signalling processes that exist among the mobile node, it's home agent and it's correspondent node (a node from outside the home network that tries to communicate with a mobile node) HMIPv6 comes as an extension to Mobile IPv6 to improve its performance and reduce the amount of signalling required, by employing a new node called Mobility Anchor Point (MAP) that deals with the delays resulting from signaling.		
	Packet Reordering	No	**Mobility Scope & Management Class**	Global & Host Based.
	Mobility Issue addressed	Briefly addresses the issues of packet loss, handover latency and end-end delay [6].		
	Other problems with the protocol	Cannot address issues like high signaling cost and huge energy consumption.		
Network Mobility (NEMO) [108]	**Description**	Employs the compressed mobility header to deal with the problem of signalling cost. A new node called the mobile router manages mobility services like sending binding updates to home agents etc. instead of the node itself.		
	Packet Reordering	No	**Mobility Scope & Management Class**	Local & Host Based.
	Mobility Issue addressed	Signaling cost packet loss, handover latency, and end-end delay addressed to a lesser extent [6].		
	Other problems	Doesnt address the issue of energy efficiency		
Proxy Mobile IPv6 (PMIPv6) [109,110]	**Description**	It has two important elements that perform all the mobility functions: Local Mobility Anchor (LMA) and Mobile Access Gateway (MAG).		
	Packet Reordering	No	**Mobility Scope & Management Class**	Local & Network Based.
	Mobility Issue addressed	Addresses the issues of signaling cost packet loss, handover latency ,and end-end delay to a lesser extent [6].		
	Other problems	Doesnt address the issue of energy efficiency.		
Sensor Proxy Mobile IPv6 (SPMIPv6) [111]	**Description**	Overcomes the problems of PMIPv6 like it solves the bottleneck ,and non-optimized path problems.		
	Packet Reordering	No	**Mobility Scope & Management Class**	Local & Network Based.
	Mobility Issue addressed	Addresses all the issues briefly [6].		
	Other problems	Issues not addressed efficiently.		
ClusteredSPMIP v6 (CSPMIPv6) [112]	**Description**	The problems in PMIPv6 arise because they use single LMAs (their load is not distributed). CSPMIPv6 uses clusters of MAGs and each cluster has its unique cluster head which perform all handover and signalling operations implying the load on LMAs getting balanced.		
	Packet Reordering	Yes	**Mobility Scope & Management Class**	Local & Network Based.
	Mobility Issue addressed	Addresses all the issues briefly [6].		
	Other problems	Issues not addressed efficiently.		
Overlapping Mobile Access Gateway (OMAG) [113]	**Description**	An extended version of PMIPv6. Covers inter-domain level. Utilizes pseudo-code to reduce latency caused by handovers.		
	Packet Reordering	Yes	**Mobility Scope & Management Class**	Global/Local & Network Based.

	Mobility Issue addressed	Addresses the issues of packet loss, handover latency and end-end delay to a lesser extent [6].		
	Other problems	Cannot address issues like high signaling cost and huge energy consumption.		
Constrained Application Protocol (CoAP) [114,115]	**Description**	Designed for low power and lossy networks. Excels in reducing handover latencies, signalling costs and packet loss.		
	Packet Reordering	Not required	**Mobility Scope & Management Class**	Doesnt apply.
	Mobility Issue addressed	Addresses all the issues to a moderate extent. Is better than other protocols [6].		
	Other problems	Best mobility management protocol till date.		

information with other companies and thus cause the information leak.
The researchers need to focus on developing a proper privacy preserving mechanism. To avoid these problems researches in the direction of privacy at four stages is needed: privacy at the device, privacy at communication, privacy at storage, and privacy at processing.

3. **Solutions and opportunities:** Data masking and encryption solutions have been proposed to secure sensitive data from leaking, but they suffer from the problem of increasing time delays and reducing the easy availability of original data. Hence, proper and generic privacy protection mechanisms need to be made that may include proper steps to be taken in the phases of data collection, data transit, data usage, data storage and finally data sharing.

5.7. Threats, Challenges and Solutions in Mobility.

1. **Threats caused by Mobility:** Mobile devices utilize volatile, and vulnerable wireless connections to append themselves to the internet. The lossy nature of these links gives rise to many problems like increased rate of error and decreased bandwidth [6] and since mobile devices have the tendency to join more and more networks, it tempts the attackers to push malicious software into them to accelerate the infection of the malicious code quickly.
2. **Challenges:** This mobility nature raises the need to develop mobility resilient security algorithms for IoT devices. The main challenge posed by the mobility feature is the cross-domain trust and identification. For example, when a mobile IoT device enters a new network, how the network should verify its credentials, and what permissions should it be provided with/limited to.
Also, when this mobile device tends to share the data in the new network,several things need to be done, e.g., key negotiation, data confidentiality, data integrity, protection etc. Table 5.5 illustrates some of the most important challenges related to the mobility feature of the IoT devices.
3. **Solutions and opportunities:** [102] tries to deal with the problem by making changes in mobile devices configuration as it joins a new network to comply with its new networks needs. However, this doesnt address the root cause of the problem. Moreover, a lot of mobility management protocols have been designed to deal with the issues in mobility.
Table 5.6 gives a view of their description as well as the issues they address. It is found that there is still scope of research in this direction and that mobility issues in IoT need to be taken more seriously.

5.8. Proposed solutions for the identified threats and challenges. Through the rigorous examination of the solutions given to various security threats and challenges of IoT in the previous sections, we comprehend that security professionals are trying to ease these threats. However, these studies need applicability and are still in the stage of infancy. As such, numerous issues still starve for efficient and IoT acceptable solutions.

In this section we propose some of the solutions (table 5.7) that could be taken up as research opportunities by the scholars, academicians or people of the industry to tackle the security threats and challenges arising from the intrinsic features of IoT devices.

6. A simple security mechanism for Smart Transport. In this section, we have explored the threats to availability of VANETs that form the basis of ITS. Though the threat on availability exists in all the discussed use cases of IoT, we have chosen to discuss the availability attack on VANETs as they are particularly vulnerable to such attacks given their typical characteristics of high mobility, dynamic topology, and lack of any centralized monitoring entity. The type of availability attack that we have studied here is the black-hole attack in which

Table 5.7
Proposed Solutions for the Identified Threats and Challenges

Solution	Description	Challenge addressed
Context-based permission systems	Such a system will help to refrain the environment and other devices from changing the devices behavior by recording and comparing parameters like procedure control flow, data source and devices behavior periodically.	Interdependence: Such a system will solve the threats discussed in section 5.1 as even if the attacker is successful at executing the misbehavior around the similar physical conditions like that of the normal, it is extremely hard to duplicate the exact context information.
Know your IoT network	The users need to keep a track of the devices and the permissions they give out. All the points in the network of an individual need to be well secured. Not only the data, but the installed IoT devices could be hacked.	Interdependence, Pervasiveness, Unattended, Affinity.
Use of anomaly based Intrusion Detection Systems	Such systems would note and report any anomalous behavior shown by the network in real-time and would work in a generic manner. Devices could be hacked to launch massive attacks.	Heterogeneity, Mobility.
Combination of IDS and honeypots	A real-time monitoring of the network can be performed. Such a system shall offer added security to the IoT without requiring putting extra pressure on resource-constrained devices.	Heterogeneity, Constrained.
Employ multi-layer protection i.e., edge layer, fog layer, and cloud layer arch.	Make the execution of attackers actions extremely difficult by having multi-layer protection for resource -constrained IoT devices as they cannot protect themselves.	Constrained, and Unattended. By having multi-layer protection, it wont be necessary to be physically present near the device.
Design lightweight cryptographic & authentication procedures	IoT devices cannot run heavy-weight cryptographic procedures.	Constrained
Run in-depth forensic analysis periodically	Regular forensic analysis over the organization by security professionals will save the network from breaches like Mirai, Reaper etc.	Pervasiveness, Mobility.
Flag any suspicious traffic	We keep a check on what comes inside but forget about doing the same with the outbound connections. When a ransomware enters the device, it needs to connect back to its Command and Control (C&C) to carry out the attack. If we are able to prevent this connection, ransomware will not be able to get off the ground at the first place. Therefore, any suspicious traffic must be stamped and investigated.	Pervasiveness, Unattended.
Up-to date device firmware	Attack success can be made difficult by plugging out any vulnerabilities and misconfigurations which might be used to penetrate the network by periodically updating the device firmware.	Heterogeneity, Pervasiveness, Interdependence.
Development of new lightweight encryption techniques	The data sent out by the IoT devices need to be secured but the existing encryption cannot be used as they are heavy weight and costly for application in these devices.	Constrained.
Use of Edge computing	Instead of sending the data out, the end devices can themselves perform computations; hence no encryption scheme will be required.	Constrained.

the offender ruses the sender node into forwarding all its data through it by offering the best and the shortest available route to the destination even if it doesnt even know it. Once it receives the senders traffic, it attacks by dropping the entire data. The thing that makes the black-hole attack more dangerous in smart transport environments is that even the internal and most authentic nodes can launch it, rendering the cryptographic techniques useless.

As such, a technique other than cryptography is required to eliminate these attacks. In this section, we propose the VANET Black-hole Prevention (VBP) algorithm that is explained below.

TABLE 6.1
Simulation Parameters

Parameters	Values
Simulation tool	NS2(2.35) + SUMO
No. of vehicles	51
No.of RSUs	5
Malicious behavior studied	Black-hole attack
No of attack scenarios	6 (with 2,4,6,8,10,12 attacker nodes respectively).
Speed of vehicles	Between 20m/s and 70 m/s
Size of packet	512 bytes.
Type of Antenna	Omni-directional
Type of MAC	802.11
Simulation time	30 minutes
Type of Data traffic	Constant Bit Rate (CBR)
Routing Protocol	Ad-hoc on-demand Vector (AODV)

The implementation of proposed algorithm depends on following points:

- Each time a node (Smart Vehicle) has to send data to another node in the network, it first finds the shortest path to destination node before sending data packets.
- Shortest path to destination node is calculated using two special control packets: Route Request Packet (RREQ) and Route Reply Packet (RREP).
- RREQ packet is generated and flooded by the source node, and contains the address of source and destination node.
- RREP packet is generated by the destination node when it receives RREQ packet. It is appended with the shortest path to source node (which has been calculated during the process of flooding of RREQ packet). It is sent as an acknowledgement to source node.
- Each intermediate node on receiving RREQ packet sends an RREP packet to source node if it has the shortest path to destination, otherwise it appends its address to RREQ packet and forwards it to neighbouring nodes.
- The source node on receiving RREP packets, extracts shortest path to destination node and uses this path to forward all the packets to destination node.
- Since the availability of computing resources at Road Side Unit (RSU) is high as compared to the smart vehicles, the nodes prefer to forward packets to destination via RSU (if it is in between source and destination node).
- RSUs not only acts as a high computing node in the smart vehicular network but also monitors the packets exchanged between smart vehicles.
- A list is maintained by an RSU: Blacklist(B), which contains the IDs of all those vehicles for which any malicious activity has been reported. Initially this list is empty.

The proposed security algorithm is executed by each node when it receives an RREP packet and its detailed working is explained as:

Algorithm:

Step-1: If the non-destination node initially generating an RREP packet, then report it to RSU which will put it in the blacklist (B), and discard the reply.

Step-2: Else if the node sending the RREP packet is already in the RSUs blacklist (B), then simply discard its reply and inform source node to re-initiate route request process.

Step-3: Else accept the RREP packet and forward it to the source node.

6.1. Experimental set-up and Evaluation. Our simulation settings and the parameters are listed in table 6.1. In every simulation, VBP is perfectly able to separate the attacks from the network.

The effect of the black-hole attack on throughput and Packet Delivery Ratios (PDR) are noted in figure 6.1. The attacker nodes have been chosen randomly and a realistic scenario of traffic has been created using SUMO.

Figure 6.1 demonstrates that as the number of attacker nodes increase, the throughput and the PDR

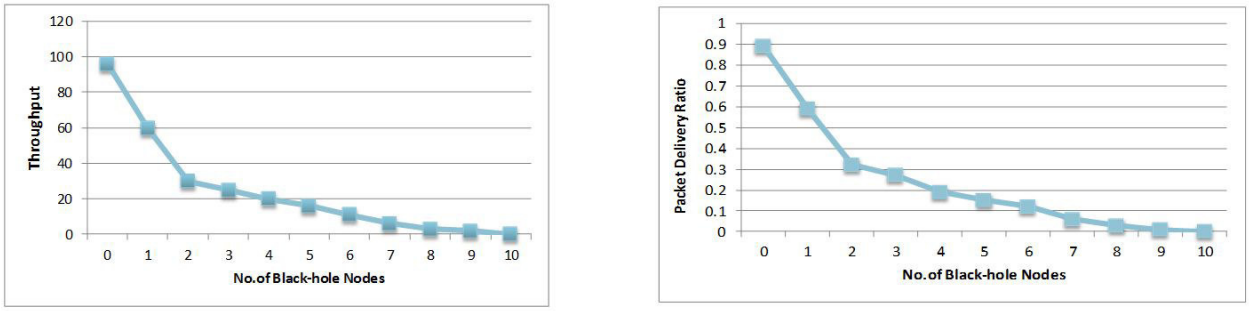


FIG. 6.1. *Effect of Black-hole attack on Throughput and Packet Delivery Ratio*

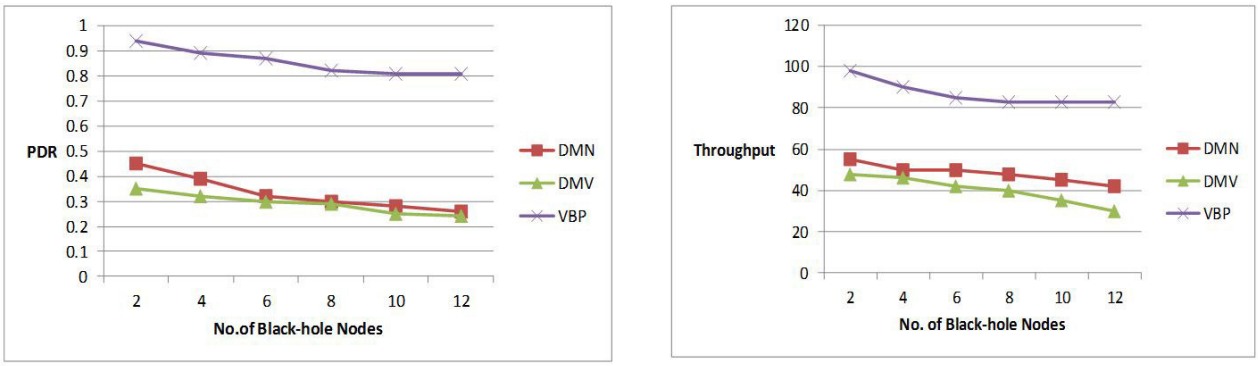


FIG. 6.2. *Comparison of Proposed VBP with DMN and DMV*

parameters decrease by a considerable amount, and touch zero when the number of attacker nodes reach to a value of ten. These figures orchestrate how badly the attacks on availability can affect the VANETs.

After analyzing the effect of black-hole attack, we have applied our VBP algorithm on smart transport network and compared it with two of the most famous malicious node detection techniques namely, Detection of Malicious Vehicle (DMV) [116] and Detection of Malicious Node (DMN) [117].The comparison charts are displayed in figure 6.2.

Figure 6.2 indicates that with VBP applied on the network, the PDR and throughput remain almost constant and high as compared to DMV and DMN under the influence of attacks. This is for the reason that DMV calculates trust values for each vehicle by assigning the job of verifiers to some other vehicles. All the verifiers work continuously in their clusters which put unrequired pressure on them, leading to wastage of resources. DMN on the other hand, uses some verifiers for the process of trust calculation. DMV and DMN give lesser values of throughput and PDR for the reason that by the time malicious nodes are detected, crucial data packets are already lost. VBP is better as it eliminates the malicious nodes right at the beginning without requiring dropping of essential packets thereby maintaining high PDR and throughput rates.

7. Conclusion and Future Work. The features peculiar to IoT devices suggest that they are helpless when it comes to securing themselves. It was observed that even the proposed security mechanisms for alleviating the possible threats suffer from a lot of problems and are not sufficient in the IoT world. Most of these security and privacy solutions are WSN inspired and thus cannot display the same degree of efficacy in IoT.

The classical security solutions are ineffective in todays IoT deployment for so many more reasons. Firstly, because of the constrained nature of the IoT devices, they do not run full-fledged operating systems. In addition, these devices have long lives-ones in which they remain unattended and unsupported by their vendors. Secondly, IoT devices dont get automated software updates because they run long after the vendor stops producing/

supporting them. Thirdly, the prevalent security procedures stem from a static perimeter defense mindset (IDS or firewall at Gateways). Since the behavior of IoT devices and their environments flip, such approaches quickly become ineffective in IoT environments. It is concluded that more effective and practical solutions are needed to address the security issues of IoT, solutions that would not put unnecessary pressure on IoT devices rendering them exhausted for performing their intended functions, solutions that do not come from a static perimeter defense mindset but rather the ones that take into account the heterogeneous, mobile and unattended natures of IoT devices. Talking about the efficacy of the security solution, if the solutions that are already available for networks like WSNs or other ad-hoc networks are employed in IoT, they would not be 100% efficient because as discussed in this article, the features of IoT are very unique.

One of the most interesting future research tendencies lies in developing more efficient Intrusion Detection and Prevention Systems to deal with the problems of IoT devices. Researchers can also find an opportunity in the creation of secure procedures for remote attestation of unattended IoT devices. There is still a lot of potential in the development of efficient context-based permission systems for dealing with issues arising from implicit dependence and in the creation of a Dynamic Analysis Simulation Platform for covering the heterogeneity in IoT firmware.

REFERENCES

[1] Fari Assaderaghi et.al, *Privacy and Security: Key Requirements for Sustainable IoT Growth*, in Symposium on VLSI Technology Digest of Technical Papers, (2017).

[2] Djamel Eddine Kouicem, Abdelmadjid Bouabdallah , Hicham Lakhlef, *Internet of things security: A top-down survey*, in Computer Networks ,Elsevier, (2018).

[3] Wei Zhou et.al, *The Effect of IoT New Features on Security and Privacy: New Threats, Existing Solutions, and Challenges Yet to Be Solved*,IEEE Internet of Things Journal, (2018).

[4] IETF, Available: https://tools.ietf.org/id/draft-feeney-t2trg-inter-network-02.html (accessed on 27 August 2018).

[5] Shen Bin et.al, *Research on Data Mining Models for the Internet of Things*, in IEEE, (2010).

[6] Muneer Bani Yassein et.al, *Mobility Management of Internet of Things: Protocols, Challenges and Open Issues*, in IEEE (2017).

[7] C. Kruger, G. Hancke, *Implementing the internet of things vision in industrial wireless sensor networks*, in Proceedings of the 12th IEEE International Conference on Industrial Informatics (2014).

[8] A. Perrig, J. Stankovic, D. Wagner, *Security in wireless sensor networks*, in Commun ACM (2004).

[9] Kewei Sha , Wei Wei , T. Andrew Yang , Zhiwei Wang, Weisong Shi, *On security challenges and open issues in Internet of Things*, in Future Generation Computer Systems, Elsevier (2018).

[10] K. Sha, J. Gehlot, R. Greve, *Multipath routing techniques in wireless sensor networks: A survey*, in Wirel. Pers. Commun. (2013).

[11] Johana A. Manrique, Johan S. Rueda-Rueda, Jesus M.T. Portocarrero, *Contrasting Internet of Things and Wireless Sensor Network from a conceptual overview*, in IEEE International Conference on Internet of Things, (2016).

[12] I. I. Initiative, *Towards a definition of the internet of things (iot)*, In Tech.Rep. (2015).

[13] G.A.Y. Wang, B. Ramamurthy, *A survey of security issues in wireless sensor networks*, in IEEE Commun. Surv. Tutor. (2006).

[14] T. Milbourn, Available:https://www.u-blox.com/en/blog/ip-versus-coap-iot-communications (accessed on August 2018).

[15] T. Bokareva, et al., *Wireless sensor networks for battlefield surveillance*, in Proceedings of Land Warfare Conference (2006).

[16] L. Larkey, L. Bettencourt, A. Hagberg, *In-situ data quality assurance for environmental applications of wireless sensor networks*, in Tech. Rep. Report LAUR-06-1117, Los Alamos National Laboratory, (Oct. 2006).

[17] Fadele Ayotunde Alaba, Mazliza Othman, Ibrahim Abaker Targio Hashem, Faiz Alotaibi, *Internet of Things security: A survey*,in Journal of Network and Computer Applications, Elsevier, (2017).

[18] H. Chan, A. Perrig, D. Song, *Random key predistribution schemes for sensor networks*, in: Proceedings of ACM CCS03, (2003).

[19] W. Du, J. Deng, Y. Han, P. Varshney, *A pairwise key pre-distribution scheme for wireless sensor networks*, in: Proceedings of ACM CCS03, (2003).

[20] M. Anita, R. Geetha, E. Kannan, *A novel hybrid key management scheme for establishing secure communication in wireless sensor networks*, in Wirel. Pers. Commun (2015).

[21] D. Liu, P. Ning, R. Li, *Establishing pairwise keys in distributed sensor networks*, in ACM Trans. Inf. Syst. Secur. (TISSEC) (2005).

[22] B. Bowerman, J. Braverman, J. Taylor, H. Todosow, U. Von Wimmersperg, *The vision of a smart city*, In: 2nd International Life Extension Technology Workshop, Paris, vol. 28 , (2000).

[23] R.G. Hollands, *Will the real smart city please stand up? intelligent, progressive or entrepreneurial? City*, 12(3), 303 - 320 , (2008).

[24] Tai-hoon Kim, Carlos Ramos, Sabah Mohammed, *Smart City and IoT*, in Future Generation Computer Systems, Elsevier

(2017).
[25] United Nations Population Fund (UNFPA) (2012), *Ageing in the twenty-first century: a Celebration and A Challenge*, Available: http://www.unfpa.org/publications/ageing-twenty-first-century (Accessed: August 2018).
[26] Global Health Workforce Alliance, World Health Organization (2013), *A universal truth: No health without a Workforce*, Available: http://www.who.int/workforcealliance/knowledge/resources/hrhreport2013/en/ (Accessed: August 2018).
[27] Haider Mshali, Tayeb Lemlouma , Maria Moloney, Damien Magoni, *A survey on health monitoring systems for health smart homes*, in International Journal of Industrial Ergonomics, Elsevier, vol:66, pp:26-56 (2018).
[28] Kirsten K. B. Peetoom, Monique A. S. Lexis, Manuela Joore, Carmen D. Dirksen, Luc P. De Witte, *Literature review on monitoring technologies and their outcomes in independently living elderly people*, in Disabil Rehabil Assist Techno, UK, (2014).
[29] Van der Gaag, N. De potentiele, beroepsbevolking , in de Europese Unie: van groei naar krimp. Den Haag/Heerlen: Statistics Netherlands [CBS] (2012).
[30] Van Duin CG J. Bevolkingsonderzoek 2010-2060, in sterkere vergrijzing, langere levensduur. Den Haag/Heerlen: Statistics Netherlands [CBS] (2010).
[31] W. AL-mawee, *Privacy and Security Issues in IoT Healthcare Applications for the Disabled Users a Survey*, Western Michigan University, 1903 W Michigan Ave, Kalamazoo, MI 49008, USA, (2012) Masters thesis .
[32] T.D.J. Clements-Croome, *What do we mean by intelligent buildings?*, in Automation in Construction 6 (1997) 395-399.
[33] W.M. Kroner, *An intelligent and responsive architecture*, in Automation in Construction 6,381393 (1997).
[34] M. Wigginton, J. Harris, *Intelligent Skin*, in Architectural Press, Oxford, UK, (2002).
[35] Terence K.L. Huia, R. Simon Sherratt, Daniel Daz Snchez, *Major requirements for building Smart Homes in Smart Cities based on Internet of Things technologies*, in Future Generation Computer Systems, Elsevier (2016).
[36] Syed Rameem Zahra, *MNP: malicious node prevention in vehicular ad hoc networks*, in International Journal of Computer Networks and Applications (2018).
[37] Debasis Bandyopadhyay, Jaydip Sen, *Internet of Things: Applications and Challenges in Technology and Standardization*, in Wireless Pers Commun,Springer (2011).
[38] M.N. Mejri , J. Ben-Othman , M. Hamdi, *Survey on vanet security challenges and possible cryptographic solutions*, in Veh. Commun. 1 (2), 53-66 (2014).
[39] M. Brettel, N. Friederichsen, M. Keller, and M. Rosenberg, *How virtualization, decentralization and network building change the manufacturing landscape: An industry 4.0 perspective*, in International Journal of Mechanical, Industrial Science and Engineering, vol. 8, no. 1, pp. 37-44 (2014).
[40] J. Wan, D. Zhang, Y. Sun, K. Lin, C. Zou, and H. Cai, *VCMIA: a novel architecture for integrating vehicular cyber-physical systems and mobile cloud computing*, in Mobile Networks and Applications, vol. 19, no. 2, pp. 153-160 (2014).
[41] Baotong Chen, Jiafu Wan, Lei Shu, Peng Li, Mithun Mukherjee and Boxing Yin, *Smart Factory of Industry 4.0: Key Technologies, Application Case, and Challenges*, in IEEE Access (2017).
[42] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, *Edge computing: Vision and challenges*, in IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637-646 (2016).
[43] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, *Industry 4.0*, in Business & Information Systems Engineering, vol. 6, no. 4, pp.239-242 (2014).
[44] E. Commission, *Europe 2020: A Strategy for smart, sustainable and inclusive growth*, Working paper [COM (2010) 2020], (2010).
[45] J. Holdren, T. Power, G. Tassey, A. Ratcliff, and L. Christodoulou, *A National strategic plan for advanced manufacturing*, in US National Science and Technology Council, Washington, DC (2012).
[46] A.-R. Sadeghi, C. Wachsmann, M. Waidner, *Security and privacy challenges in industrial internet of things*, in: 2015 52nd ACM/EDAC/IEEE on Design Automation Conference (DAC), IEEE, pp. 1-6 (2015).
[47] Tianlong Yu, Vyas Sekar, Srinivasan Seshan, Yuvraj Agarwal, Chenren Xu, *Handling a trillion (unfixable) flaws on a billion devices:Rethinking network security for the Internet-of-Things*, in HotNets, ACM (2015).
[48] Jia, Yunhan Jack, et al., *ContexIoT: Towards Providing Contextual Integrity to Appified IoT Platforms*, in Network and Distributed System Security Symposium , pp. 1-15 (2017).
[49] Luca Mainetti, Luigi Patrono, Andrea Secco, and Ilaria Sergi, *An IoT-aware AAL System for Elderly People*, in International Multidisciplinary Conference on Computer and Energy Science (SpliTech), IEEE (2016).
[50] Ali (2015), *Internet of things security report[Online]*, Available: https://jaq.alibaba.com/community/art/show?articleid =195 (accessed july 2018).
[51] JD. (2015). *Joylink*, Available: http://smartdev.jd.com/ (accessed july 2018).
[52] Yang Yu, *BadTunnel:NetBIOS Name Service spoofing over the Internet*, in Tencents Xuanwu Lab (2016).
[53] File Manipulation, Available: https://capec.mitre.org/data/definitions/165.html (accessed august 2018).
[54] File Discosure, Available: https://teamultimate.in/full-path-disclosure-attack-explained-tutorial/ (accessed august 2018).
[55] Command Injection , Available: http://cwe.mitre.org/data/definitions/77.html (accessed august 2018).
[56] Code Injection, Available: https://www.owasp.org/index.php/Code_Injection (accessed august 2018).
[57] Unserialize, Available: http://php.net/manual/en/function.unserialize.php (accessed august 2018).
[58] Andrei Costin, Jonas Zaddach, Aurelien Francillon and Davide Balzarotti, *A Large-Scale Analysis of the Security of Embedded Firmwares*, in 23rd USENIX Security Symposium (2014).
[59] Drew Davidson,Benjamin Moench, Somesh Jha, Thomas Ristenpart, *FIE on Firmware: Finding Vulnerabilities in Embedded Systems using Symbolic Execution*, in 22nd USENIX Security Symposium (2013).
[60] Jonas Zaddach, Luca Bruno, Aurelien Francillon and Davide Balzarotti, *Avatar: A Framework to Support Dynamic*

Security Analysis of Embedded Systems' Firmwares, in Internet Society (2014).
[61] Daming D. Chen, Manuel Egele, Maverick Woo, and David Brumley, *Towards Automated Dynamic Analysis for Linux-based Embedded Firmware*, in Internet Society (2016).
[62] Franqueira, Virginia Nunes Leal and Wieringa, Roel J, *Role Based Access Control in Retrospect*, in Central Lancashire online Knowledge (2012).
[63] OASIS, *eXtensible Access Control Markup Language (XACML) Version 3.0*
[64] Sergio Gusmeroli a, Salvatore Piccione, Domenico Rotondi, *capability-based security approach to manage access control in the Internet of Things*, in Mathematical and Computer Modelling,Elsevier 58, 1189-1205 (2013).
[65] Ki-Wook Kim , Youn-Hee Han, Sung-Gi Min, *An Authentication and Key Management Mechanism for Resource Constrained Devices in IEEE 802.11-based IoT Access Networks*, in Sensors (2017).
[66] Zhao, Lu, et al., *ARMor: fully verified software fault isolation*, In Proceedings of the International Conference on Embedded Software IEEE, 289-298 (2011).
[67] Schulz, Patrick Koeberl Steffen, Ahmad-Reza Sadeghi, and Vijay Varadharajan, *Trustlite: A security architecture for tiny embedded devices*, In EuroSys. ACM, pp:1-14 (2014).
[68] Guo, Fuchun, et al. , *CP-ABE With Constant-Size Keys for Lightweight Devices*, In IEEE Transactions on Information Forensics & Security 9.5, pp. 763-771 (2014).
[69] Fan, Hongfei, et al., *An ultra-lightweight white-box encryption scheme for securing resource-constrained IoT devices*, In Conference on Computer Security Applications ACM, pp.16-29 (2016).
[70] Buchmann, Johannes, et al., *High-performance and lightweight lattice-based public-key encryption*, In Proceedings of the 2nd ACM 10 International Workshop on IoT Privacy, Trust, and Security. ACM, pp. 2-9 (2016).
[71] Rauter, Tobias, N. Kajtazovic, and C. Kreiner, *Privilege-Based Remote Attestation: Towards Integrity Assurance for Lightweight Clients*, in ACM Workshop on IoT Privacy, Trust, and Security .ACM,pp. 3-9 (2015).
[72] B. Hayes, *Cloud Computing*, Commun. ACM, vol. 51, no. 7, pp. 9-11 (2008).
[73] N. C. Luong et al., *Data collection and wireless communication in Internet of Things (IoT) using economic analysis and pricing models: A survey*, in IEEE Commun. Surveys Tuts., vol. 18, no. 4, pp. 25462590, 4th Quart. (2016).
[74] R. Roman, J. Lopez, and M. Manbo, *Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges*, Future Gener. Comput. Syst., vol. 78, pp. 680-698 (Jan. 2018).
[75] Mithun Mukherjee, Rakesh Matam, Lei Shu, Leandros Maglaras, Mohamed Amine Ferrag, Nikumani Choudhury, Vikas Kumar, *Security and Privacy in Fog Computing: Challenges*, in IEEE Access (2017).
[76] Multitude threat, *Available: https://www.techrepublic.com/article/ddos-attacks-increased-91-in-2017-thanks-to-iot/ (accessed august 2018)*.
[77] Mirai, Available: https://blog.cloudflare.com/inside-mirai-the-infamous-iot-botnet-a-retrospective-analysis/ (accessed august 2018).
[78] Mirai, Available: https://www.csoonline.com/article/3258748/security/the-mirai-botnet-explained-how-teen-scammers-and-cctv-cameras-almost-brought-down-the-internet.html(accessed august 2018).
[79] Mirai, Available:http://effortlessoffice.com/mirai-botnet-attack-costs-companies-hundreds-of-millions/(accessed august 2018).
[80] Mirai, Available: https://www.corero.com/blog/797-financial-impact-of-mirai-ddos-attack-on-dyn-revealed-in-new-data.html(accessed august 2018).
[81] Reaper, Available: https://krebsonsecurity.com/2017/10/reaper-calm-before-the-iot-security-storm/(accessed august 2018).
[82] Reaper, Available: https://www.wired.com/story/reaper-iot-botnet-infected-million-networks/(accessed august 2018).
[83] Reaper, Available: https://www.fortinet.com/blog/threat-research/reaper-the-next-evolution-of-iot-botnets.html(accessed august 2018).
[84] Reaper, Available: https://www.zdnet.com/article/reaper-botnet-experts-reassess-size-and-firepower/(accessed august 2018).
[85] Sam Edwards, Ioannis Profetis, *Hajime: Analysis of a decentralized internet worm for IoT devices*, in Rapidity Networks Security Research Group (2016).
[86] Hajime, Available: https://www.corero.com/blog/882-hajime-botnet-scanning-for-vulnerable-mikrotik-routers.html (accessed august 2018).
[87] Hajime, Available: https://indianexpress.com/article/technology/tech-news-technology/mysterious-hajime-malware-infecting-iot-network-globally-4631556/ (accessed august 2018).
[88] Wannacry, Available: http://www.hendonpub.com/resources/article_archive/results/details?id=5903 (accessed august 2018).
[89] Perseria, Available: https://www.theinquirer.net/inquirer/news/3009839/persirai-mirai-a-like-malware-is-your-latest-iot-security-worry(accessed august 2018).
[90] Perseria, Available: https://securityaffairs.co/wordpress/59900/malware/persirai-iot-botnets.html(accessed august 2018).
[91] Bushra Zaheer Abbasi, Munam Ali Shah, *Fog Computing: Security Issues, Solutions and Robust Practices*, in Proceedings of the 23rd International Conference on Automation & Computing, University of Hudders field, Hudders field, UK, 7-8 (September 2017).
[92] Zhang, Congyingzi, and R. Green, *Communication security in internet of thing: preventive measure and avoid DDoS attack over IoT network*, Symposium on Communications & NETWORKING Society for Computer Simulation International, pp. 8-15 (2015).
[93] Guan, Le, et al., *TrustShadow: Secure execution of unmodified applications with ARM trustzone*, in Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services, ACM (2017).
[94] K. E. Defrawy, A. Francillon, D. Perito, and G. Tsudik, *SMART: Secure and minimal architecture for (establishing a*

dynamic) root of trust, in Network. & Distribution. System. Security Symp (2012).
[95] Copos, Bogdan, et al., *Is Anybody Home? Inferring Activity From Smart Home Network Traffic*, In Security and Privacy Workshops IEEE, pp. 245-251 (2016).
[96] K. Ki-Sik, S. MoonBae, Kwang Jin, P.A.R.K. and C.S Hwang, *A comparative analysis on the signaling load of Mobile IPv6 and Hierarchical Mobile IPv6: Analytical approach*, in IEICE Transactions on Information and Systems, 89(1), 139-149 (2006).
[97] Wikipedia contributors, Packet loss (2016), Available: https://en.wikipedia.org/wiki/Packet_loss (accessed july 2018).
[98] K. S. Kong, W. Lee, Y. H. Han and M. K. Shin, *Handover latency analysis of a network-based localized mobility management protocol*, In IEEE International Conference on Communications, pp. 5838-5843 (2008).
[99] Wikipedia contributors, End-to-end delay, Available: https://en.wikipedia.org/wiki/End-to-end_delay (accessed july 2018).
[100] Y. Xu, *Minimize end-to-end delay through cross-layer optimization in multi-hop wireless sensor networks* (2010).
[101] H. Y. Hwang, S. J. Kwon, Y. W. Chung, D. K. Sung and S. Park, *Modeling and Analysis of an Energy-Efficient Mobility Management Scheme in IP-Based Wireless Networks*, in Sensors, 11(12), pp.11273-11294 (2011).
[102] Samsung SmartThings, Available: https://www.smartthings.com/ (accessed july 2018).
[103] C. Perkins, *IP mobility support for IPv4*, No. RFC 3344 (2002).
[104] D. Johnson, C. Perkins and J. Arkko, *Mobility support in IPv6*, No. RFC 3775 (2004).
[105] N. Jora, *Mobile IP and Comparison between Mobile IPv4 and IPv6. Journal of Network Communications and Emerging Technologies*, in (JNCET) www. jncet. org, 2(1) (2015).
[106] T. Narten, W.A. Simpson, E. Nordmark, and H. Soliman, *Neighbor discovery for IP version 6 (IPv6)*, (2007).
[107] H. Soliman, L. Bellier and K.E. Malki, *Hierarchical mobile IPv6 mobility management (HMIPv6)*, (2005).
[108] V. Devarapalli, R. Wakikawa, A. Petrescu and P. Thubert, *Network mobility (NEMO) basic support protocol*, No. RFC 3963 (2004).
[109] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury and B. Patil, *Proxy mobile ipv6*, No. RFC 5213 (2008).
[110] I. Joe & H. Lee, *An efficient inter-domain handover scheme with minimized latency for PMIPv6*, In Computing, Networking and Communications, in (ICNC). Proceedings of International Conference on IEEE, pp.332-336 (2012).
[111] M.M. Islam and E.N. Huh, *Sensor proxy mobile IPv6(SPMIPv6)-A novel scheme for mobility supported IP-WSNs*, in Sensors, 11(2), pp.1865-1887 (2011).
[112] A.J. Jabir, S.K. Subramaniam, Z.Z. Ahmad and N.A. Hamid, *A cluster-based proxy mobile IPv6 for IP-WSNs*, EURASIP Journal on Wireless Communications and networking, (1), pp.1-17 (2012).
[113] S. Ro and V.H. Nguyen, *Inter-domain mobility support in Proxy Mobile IPv6 using overlap function of mobile access gateway*, in Wireless Networks, 21(3), pp.899-910 (2015).
[114] Z. Shelby, K. Hartke and C. Bormann, *The constrained application protocol (CoAP)*, in No. RFC 7252 (2014).
[115] S.M. Chun, H.S. Kim, and J.T. Park, *CoAP-Based Mobility Management for the Internet of Thing*, in Sensors, 15(7), pp.16060-16082 (2015).
[116] Ameneh Daeinabi, Akbar Ghaffarpour Rahbar, *Detection of malicious vehicles (DMV) through monitoring in vehicular Ad-hoc Networks*, Springer, pp. 325-338 (2013).
[117] Uzma Khan, Shikha Agarwal, Sanjay Silakari, *Detection of Malicious Nodes (DMN) in Vehicular Ad-Hoc Networks*, Elsevier, pp 965- 972 (2015).

Edited by: Shajulin Benedict
Received: March 31, 2019
Accepted: August 30, 2019

DOI 10.12694/scpe.v20i3.1545
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 3, pp. 485–494. http://www.scpe.org

ENSEMBLE SPATIO-TEMPORAL DISTANCE NET FOR SKELETON BASED ACTION RECOGNITION

NAVEENKUMAR M. AND DOMNIC S.*

Abstract. With the recent developments in sensor technology and pose estimation algorithms, skeleton based action recognition has become popular. This paper proposes a deep learning framework for action recognition task using ensemble learning. We design two subnets to capture spatial and temporal dynamics of the entire video sequence, referred to as $Spatial-distance\ Net$ ($SdNet$) and $Temporal-distance\ Net$ ($TdNet$) respectively. More specifically, $SdNet$ is a Convolutional Neural Network based subnet to capture spatial dynamics of joints within a frame and $TdNet$ is a long short term memory based subnet to exploit temporal dynamics of joints between frames along the sequence. Finally, two subnets are fused as one ensemble network, referred to as $Spatio-Temporal\ distance\ Net$ ($STdNet$) to explore both spatial and temporal information. The efficacy of the proposed method is evaluated on two widely used datasets, UTD MHAD and NTU RGB+D, and the proposed $STdNet$ achieved 91.16% and 82.55% accuracies respectively.

Key words: Human action recognition, Skeleton maps, spatio-temporal distance net, CNN, LSTM

AMS subject classifications. 68M14, 92B20

1. Introduction. Action recognition is a hot research topic in the field of computer vision. It has various practical applications such as video surveillance, human-computer interaction, elderly care monitoring, smart homes, etc. The earlier studies have been investigated the action recognition task using RGB sensors. When low cost 3D sensors are available in the market, action recognition using depth data has become popular. Depth data is invariant to illumination changes compared with RGB data. In the seminal work, Shatton et al. [24] presented an approach to get skeleton joints from depth data in real time. It has generated a renewed interest in the research community to use of skeleton data for action recognition. Moreover, Skeleton map is invariant to viewpoints or appearances compared with depth map, thus suffering less intra-class variance [36].

In the past decade, multiview learning based methods [26] have achieved state of the art performance in the field of computer vision. In multiview learning, different views (features) are obtained either from multiple sources or from a single source. The traditional methods have been focused on designing hand crafted features for action recognition task [1] [27]. Due to limited representation power of hand crafted features, they often fail on large datasets. Deep multiview based methods [19] [32] have received much attention in the recent years. Most of these methods use a single type of deep network for action recognition [33]. Unlike these methods, this paper proposes a ensemble network using multiple convolutional neural networks (CNN) and multiple long short term memory (LSTM) neural networks.

The contributions of this paper are three fold. First, We design two subnets to capture spatial and temporal dynamics of the entirety sequence, referred to as $Spatial-distance\ Net$ ($SdNet$) and $Temporal-distance\ Net$ ($TdNet$) respectively. Specifically, $SdNet$ is a CNN based network and $TdNet$ is a LSTM based network. Second, the two subnets are fused as one ensemble network ($STdNet$) for action recognition task. Last, the performance of the proposed method is investigated by conducting extensive experiments on two benchmark datasets and detailed analysis is reported. The rest of the paper is organized as follows. Section 2 gives a brief overview of related works in the literature and the proposed method is presented in Sec. 3. The experimental results are described in Sec. 4. The conclusions are drawn in the final section.

2. Related Work. In this section, we briefly review the literature related to our work i.e. skeleton based approaches for action recognition. Existing literature about the skeleton based action recognition can be classified into two types: Handcrafted feature based methods (Traditional methods), Deep learning based methods.

2.1. Handcrafted feature based methods. These methods can be classified into three categories: (i) joint based, (ii) mined joint based and (iii) dynamics based methods. The joint based methods capture the correlation between the joints for action recognition task. For example, Mller et al. [20] introduce a class of

*Department of Computer Applications, NIT, Tiruchirappalli, (mnaveenmtech@gmail.com, domnic@nitt.edu)

boolean features expressing geometric relations between body points of a pose. Kerola et al. [14] has employed a graph based approach for action recognition. In this work, an action sequence is represented as spatio-temporal graph and edge weights are calculated based on the pair wise distances. Hussein et al. [13] has used the covariance matrix of skeleton sequence as feature descriptor for action recognition and the multiple covariance matrices are generated to capture the relation between joint movement and time. Mined joint based methods try to learn which body parts are discriminative for action recognition. In paper [4], a genetic algorithm is proposed to identify informative joints for a specific class. Then, K-means algorithm is employed for action recognition task. In work [29], skeleton joints are grouped into five body parts. Then, data-mining techniques are applied to obtain distinctive poses in spatial domain and temporal domain. Support Vector Machine is employed for action classification. In dynamics based methods, the temporal dynamics are captured from the 3D trajectories of the skeleton action sequence for action classification task. Dynamic based approaches use linear dynamical systems(LDS) [2] or hidden Markov models (HMM) or mixed approaches [22] for modeling of actions. Handcrafted features are having limited representation capability and hence they often fail on large datasets.

2.2. Deep learning based methods. There are two types of deep learning models received much attention for action recognition task. They are (i) Recurrent Neural Networks (RNNs) and (ii) Convolutional Neural Networks (CNNs).

Different RNN based structures such as hierarchical RNN [7], spatio-temporal long short-term memory (LSTM) [17], part-aware LSTM [23], spatio-temporal attention based RNN [25] and two stream RNN [30] have been proposed to learn discriminative features from skeleton data for action recognition. The above methods concatenate the coordinates of joints at each time step before applying RNN based methods. Thus, spatial geometrical relations among different joints are lost in this pre-processing stage.

In contrast, CNNs directly extract information from texture images which are encoded from skeleton sequences. Different CNN based methods are proposed for skeleton based action recognition. In [6], the 3D (x,y,z) coordinates of joints in skeleton action sequence are mapped into red, blue and green values respectively. Hence, the skeleton action sequence is converted into the color image, and the action recognition problem is converted to image classification problem, and then the powerful image classifiers such as Convolution Neural Networks (CNN) are employed for action recognition. Due to the small size of the converted color images, it is difficult to fine-tune the existing CNN. In the work [34], the joint trajectories are extracted and encoded into color images by using hue, saturation, and values. The encoded trajectories are used in CNN as inputs for action classification. The joint trajectories capture temporal variations, but fail to extract structural dynamics within a frame in the action sequence. In this context, Li et al. [16] proposed four Joint Distance Maps based on the pairwise distances of skeleton joints within the frame and the CNN was adopted for action recognition. But this method fails to distinguish some actions, which are having similar distance variations i.e. drawcircleclockwise and drawcirclecounter clockwise, due to the loss of local temporal information. To address this issue, this paper proposes an ensemble network, which is formed using CNN and LSTM based networks (SdNet and TdNet), to capture spatial and temporal dynamics of joints along the sequence. It is note that the success of an action recognition task is dependent on how effectively a model captures the spatial and temporal dynamics from the action sequences to achieve higher recognition accuracy.

3. Proposed Method. In this section, we first introduce the some necessary backgrounds. Then, we present two phases, Feature Extraction and Action Representation, of the proposed method. Finally, the action classification phase is discussed.

3.1. Preliminaries. Recurrent Neural Networks (RNN) are designed to model sequential problems. RNN has it's internal memory and can store information about past computations. It allows RNN to exhibit dynamic temporal behaviour. In theory, they can handle arbitrary length sequences, but in practice, RNNs have trouble to model long term sequences due to vanishing/exploding gradients problem. To overcome this problem, Long Short Term Memory (LSTM) [11] is proposed. The basic structure of LSTM unit is shown in Fig. 3.1. From the Fig 3.1, X_t is the input to the LSTM at time step t. I_t, F_t, G_t and O_t are the internal structures of input, forget, cell candidate and output gates of LSTM. I_t, F_t, G_t and O_t are defined at time step t as stated in Equations 3.1 to 3.4 respectively. The cell state (C_t) and hidden state of LSTM (H_t) are updated as stated in

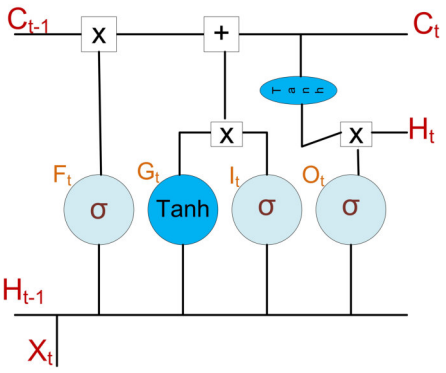


FIG. 3.1. *An LSTM unit. O_t, F_t and I_t are output, forget and input gates of LSTM. '+' and '×' are the element wise addition and multiplication respectively.*

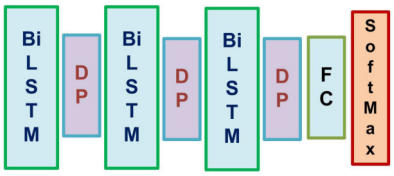


FIG. 3.2. *Block diagram of a BaseNet. BiLSTM, DP, and FC represent the bidirectional LSTM, dropout and fully connected layers.*

Equations 3.5 and 3.6 respectively.

$$I_t = \sigma(W_{IX}X_t + W_{IH}H_{t-1} + b_I) \tag{3.1}$$

$$F_t = \sigma(W_{FX}X_t + W_{FH}H_{t-1} + b_F) \tag{3.2}$$

$$O_t = \sigma(W_{OX}X_t + W_{OH}H_{t-1} + b_O) \tag{3.3}$$

$$G_t = Tanh(W_{GX}X_t + W_{GH}H_{t-1} + b_G) \tag{3.4}$$

$$C_t = F_tC_{t-1} + I_tG_t \tag{3.5}$$

$$H_t = O_tTanh(C_t) \tag{3.6}$$

where W and b represent the weight matrix and bias respectively.

3.2. BaseNet. The proposed *BaseNet* is a multi-layer LSTM network. The backbone of our *BaseNet* contains three bidirectional LSTM (Bi-LSTM) layers as shown in figure 3.2. The dropout (DP) is employed between two bi-LSTM layers to reduce the overfitting problem in training our *BaseNet*. Finally a fully connected layer (FC) with softmax activation function is used for action classification task.

3.3. Spatial-distance Net (SdNet). We design the *Spatial − distance Net* (*SdNet*) to explore the spatial dynamics of joints of a entirety sequence. The proposed *SdNet* contains four CNN as shown in Fig. 3.3. With the motivation of the work [16], *SdNet* employs pair wise distances constructed in 3D space and three 2D

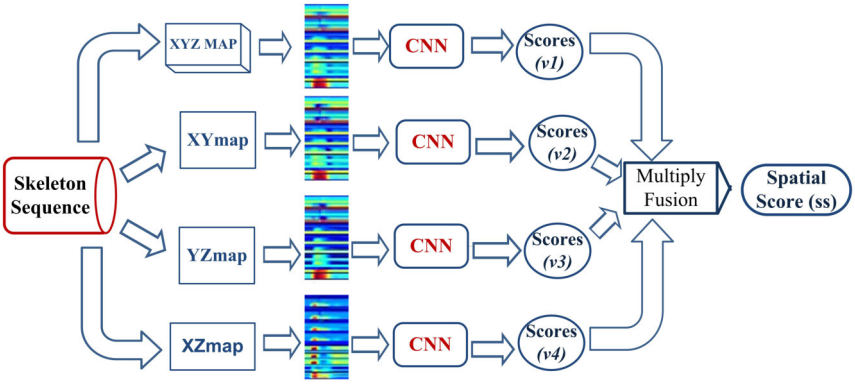


FIG. 3.3. *Spatial-distance Net (SdNet)*

orthogonal spaces as shown in Fig 3.3. Unlike [16], this paper proposes $TdNet$ to explore temporal dynamics between frames along the sequence as explained in Section 3.4

The four features in the spatial domain are referred to $XYZmap, XYmap, YZmap$ and $XZmap$. To deal with actions that contains human to human interaction, every action is assumed to be performed by two subjects (main subject and auxiliary subject). If an action sequence contains only one subject, a shadow subject is copied from main subject [5]. Suppose an action sequence A contains M skeleton frames and each skeleton frame contains $2N$ joints, where N joints are related to main subject and other N joints are for auxiliary subject. A $= \{F_1, ...F_M\}$, where F represents a frame and $F_i = \{J_1^i, ...J_{2N}^i\}$. The J_j^i represents the 3D coordinates (x, y, z) of j^{th} joint of i^{th} frame. The 2D orthogonal projections of 3D Joint J are referred as Jxy, Jyz and Jxz. The four spatial features are constructed as stated in Eqs. (3.7) to (3.10).

$$XYZmap = \{dist_{3D}(J_i^f, J_j^f)|i, j = 1.., 2N; i \neq j; f = 1.., M\} \tag{3.7}$$

$$XYmap = \{dist_{2D}(Jxy_i^f, Jxy_j^f)|i, j = 1.., 2N; i \neq j; f = 1.., M\} \tag{3.8}$$

$$YZmap = \{dist_{2D}(Jyz_i^f, Jyz_j^f)|i, j = 1.., 2N; i \neq j; f = 1.., M\} \tag{3.9}$$

$$XZmap = \{dist_{2D}(Jxz_i^f, Jxz_j^f)|i, j = 1.., 2N; i \neq j; f = 1.., M\} \tag{3.10}$$

where f represents the frame number, J refers (x,y,z) coordinates of joint, Jxy refers (x,y) coordinates of the joint and so on. $dist_{3D}()$ is the Euclidean distance of two points in Euclidean 3-space where as $dist_{2D}()$ is the Euclidean distance of two points in Euclidean 2-space. suppose, $r = (r_1, r_2, ..., r_n)$ and $s = (s_1, s_2...., s_n)$ are two points in Eucledean n-space, the $dist_{nD}()$ is calculated as:

$$dist_{nD}(r, s) = \sqrt{(s_1 - r_1)^2 + (s_2 - r_2)^2 + ...(s_n - r_n)^2} \tag{3.11}$$

For an action A, when the distances calculated for a single frame are arranged in a single column, four matrices are generated for four spatial features respectively. Each matrix of size $(2N^2 - N) \times M$. Since number of frames (M) is vary from sequence to sequence, feature matrices do not contain fixed number of columns for all the sequences in the training set. To avoid this problem and produce matrices with fixed number of columns M', bi-linear interpolation is used to resize the spatial feature matrix from $(2N^2 - N) \times M$ to $(2N^2 - N) \times M'$. Then, these feature matrices are encoded into gray images as stated in equation 3.12:

$$grayimage = \frac{FM - min(FM)}{max(FM) - min(FM)} * 255 \tag{3.12}$$

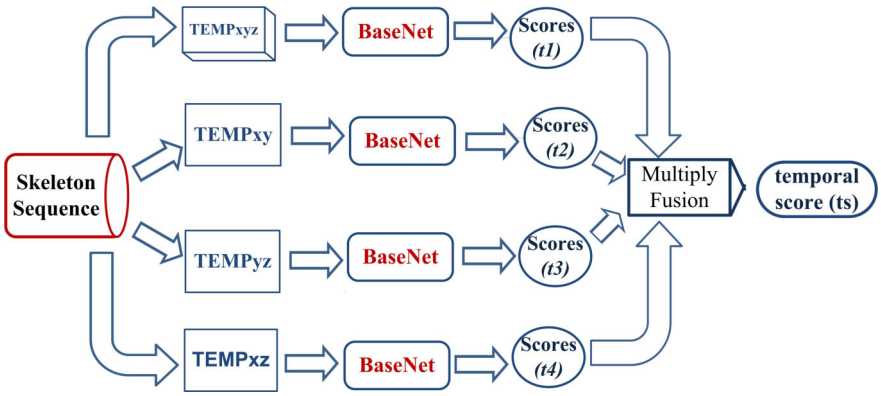


FIG. 3.4. *Temporal-distance Net (TdNet)*

where FM is a feature matrix. $min(FM)$ and $max(FM)$ are the minimum and maximum values of FM. The gray images are encoded into color texture images using Jet colorbar [16] to exploit spatial information using pretrained CNN models. This paper adopts multiplication fusion [16] to calculate the spatial score (ss) of $Sdnet$. Suppose $v1, v2, v3$, and $v4$ are the vectors related to scores of four CNNs, spatial score (ss) for action A is calculated as stated in Eq. (3.13):

$$spatial\ score\ (ss)\ for\ action\ A = (v1\ \diamond\ v2\ \diamond\ v3\ \diamond v4\ ) \tag{3.13}$$

where $\diamond$ represents the element wise multiplication.

3.4. Temporal-distance Net (TdNet). The proposed $Temporal-distance\ Net\ (TdNet)$ contains four $BaseNet$ as shown in Fig. 3.4. Four temporal features, referred to $TEMPxyz$, $TEMPxy$, $TEMPyz$ and $TEMPxz$ are constructed as stated in Eqs (3.14) to (3.17):

$$TEMPxyz = \{dist_{3D}(J_i^f, J_i^{f+1}) | i = 1.., 2N; f = 1.., M-1\} \tag{3.14}$$

$$TEMPxy = \{dist_{2D}(Jxy_i^f, Jxy_i^{f+1}) | i = 1.., 2N; f = 1.., M-1\} \tag{3.15}$$

$$TEMPyz = \{dist_{2D}(Jyz_i^f, Jyz_i^{f+1}) | i = 1.., 2N; f = 1.., M-1\} \tag{3.16}$$

$$TEMPxz = \{dist_{2D}(Jxz_i^f, Jxz_i^{f+1}) | i = 1.., 2N; f = 1.., M-1\} \tag{3.17}$$

In the context of the $TEMPmap$ feature, the distances calculated for two consecutive frames are arranged in a single column (i.e. each column as time step (X_t)). As a result, a matrix is formed of size $2N \times (M-1)$ for each $TEMPmap$ feature. Then, bi-linear interpolation is used to resize the $TEMPmap$ from $2N \times (M-1)$ to $2N \times M'$. The output of $TdNet$ for an action A is temporal score (ts). Suppose t1, t2, t3 and t4 are the score vectors of four $BaseNet$, (ts) is defined as stated in Eq. 3.18. It is note that the spatial score (ss) and temporal score (ts) vectors are in same size.

$$temporal\ score\ (ts)\ for\ action\ A = (t1\ \diamond\ t2\ \diamond\ t3\ \diamond t4\ ) \tag{3.18}$$

where $\diamond$ represents the element wise multiplication.

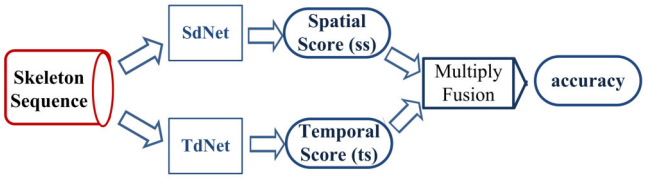


FIG. 3.5. *Spatio Temporal distance Net (STdNet)*

3.5. Spatio Temporal distance Net (*STdNet*). *STdNet* is a ensemble network, consists of CNN (*SdNet*) and LSTM (*TdNet*) based subnets, to explore spatial and temporal dynamics. Specifically, the proposed two nets (*SdNet* and *TdNet*) will be trained independently, with cross-entropy as the cost function. Suppose the training set contains K number of classes, the cross entropy is calculated as shown in Eq. 3.19:

$$cross\ entropy\ loss = -\sum_{i=1}^{K} y_i(log(p_i) \tag{3.19}$$

where y_i and p_i are true and predicted probabilities of class i. After training of *SdNet* and *TdNet*, the ensemble of these two networks, referred to Spatio Temporal distance Net (STdNet), is constructed as shown in Fig. 3.5. The class label for an unknown test instance A is calculated as stated in Eq. 3.20:

$$Label\ of\ test\ instance\ A = Find\_Max\_index(ss\ \diamond\ ts) \tag{3.20}$$

where $\diamond$ represents the element wise multiplication and $Find\_Max\_index(.)$ is the function to find the index (class label) of maximum value.

4. Experiments. The efficacy of the proposed ensemble network (*STdNet*) is evaluated on two benchmark datasets for action recognition. The details are as follows.

4.1. Implementation details. To achieve better results, the popular CNN architecture "ResNet" [10] is fine-tuned for *SdNet*. The matlab implementation of resnet50 (pretrained) model is used for all the experiments. The initial learning rate is 0.001 and batch size is set to 32. Fifteen maximum training cycles are fixed for all the experiments. The network weights are learned using backpropagation with stochastic gradient descent with momentum value set to 0.9. For *BaseNet*, the base learning rate is 0.001 and the drop out rates are set to 0.3, 0.3 and 0.5 for three dropout layers respectively to prevent overfitting. All the experiments are carried out using NVIDIA Titan XP graphics card. The number of epochs are set to 200 and mini-batch size is 128 for all the experiments using *BaseNet*.

4.2. Datasets.

4.2.1. UTD MHAD dataset. UTD MHAD dataset [3] uses a Kinect camera and a wearable inertial sensor to capture depth, skeleton joints, RGB data and inertial sensor data. The skeleton is represented by using 20 joints. It contains 27 actions, performed by 8 subjects with each one performs an action four times. As a result, 864 (27 x8 x 4 = 864) data sequences are generated. After removing 3 corrupted sequences, the total data sequences are 861. For a fair comparison, we follow the cross subject protocol proposed in the paper [3]. For cross subject evaluation, the odd subjects are used for training and even subjects are used for testing. As a result, there are 431 action sequences in the training set and 430 in the testing set. Since this data set contains less number of instances, it is not suitable for training a deep learning model. Hence, we use a transfer learning approach for this dataset. Specifically, the pre-trained models on NTU RGB+D dataset are used for transfer learning.

4.2.2. NTU RGB+D dataset. NTU RGB+D dataset [23] is the largest action recognition dataset and it uses three kinect v2 sensors to capture the depth and skeleton information. The skeleton is represented using 25 joints. There are 56,880 action sequences and more than 4 million frames in this dataset. After removing

TABLE 4.1
Recognition accuracy of SdNet, TdNet and STdNet on UTD MHAD and NTU RGB+D datasets

Feature	UTD MHAD	NTU RGB+D	
	CS (%)	CS (%)	CV (%)
XYZmap	80.93	75.53	83.23
XYmap	78.14	71.72	74.42
YZmap	76.05	71.46	71.30
XZmap	73.72	68.34	75.93
TEMPxyz	65.58	58.77	64.72
TEMPxy	64.19	54.99	64.06
TEMPyz	68.37	55.38	58.96
TEMPxz	61.40	57.40	61.46
SdNet	87.67	80.61	86.73
TdNet	74.19	66.39	73.06
STdNet	91.16	82.55	88.46

TABLE 4.2
Comparison results on UTD MHAD Dataset

	Accuracy(%)
ElC-KSVD, 2014 [37]	76.19
Kinect and Inertial, 2015 [3]	79.10
Joint trajectory maps, 2016 [34]	85.81
Joint Distance Maps, 2017 [16]	88.10
Our method (STdNet)	**91.16**

missing skeletons, the dataset contains 56,578 action sequences. This dataset is challenging in two aspects: (i) large intra class variations; (ii) view point variations. Due to large scale of this dataset, it is highly suitable for deep learning. We follow the two experimental protocols, namely cross subject and cross view protocols, proposed in paper [23]. In cross subject test, the actions pertaining to the subjects 1, 2, 4, 5, 8, 9, 13, 14, 15, 16, 17, 18, 19, 25, 27, 28, 31, 34, 35, 38 are considered for training and rest are for testing. As a result, the training set contains 40,091 samples, whereas testing set consisting of 16,487 action sequences. In cross view evaluation, the samples captured using camera 2 and 3 for training and camera 1 samples for testing.

4.3. Results of Action Recognition. Table 4.1 reports the results of proposed $SdNet$, $TdNet$ and $STdNet$. When comparing the individual spatial features, $XYZmap$ outperforms the other features on both the datasets. $STdNet$ is the result of ensemble of two networks $SdNet$and $TdNet$. $STdNet$ significantly achieves good performance than other recent works in the literature. From these results, it is concluded that both $SdNet$ and $TdNet$ have their significance to achieve recognition accuracy. Table 4.2 reports the performance of the proposed method with the state of the art methods on UTD MHAD dataset. It is noted from Table 4.2 that the works [34] [16] achieved 85.81% and 88.10% recognition accuracies respectively, which is not better than that of the proposed method, which achieves the accuracy of 91.16%. The reason is that the proposed method uses both spatial and temporal dynamics whereas the works [34] [16] used either spatial or temporal dynamics for action recognition.

Table 4.3 reports the results on the NTU RGB+D dataset. For this dataset, we compare our results with traditional methods [28] [8] [12], RNN based methods [7] [23] [25] [18] [30] [31] and CNN based methods [34] [16] [21]. The empirical results show that our $STdNet$ achieves 82.55% and 88.46% accuracies for cross subject and cross view settings respectively, which are higher than the recent existing works. Figure 4.1 depicts the individual recognition accuracies of all action classes. Among 60 classes, 36 action classes have achieved >=90% recognition rate in the cross view experimental setting whereas 24 action classes in the cross subject test. Table 4.4 shows the different action classes pertaining to specific recognition range on NTU RGB+D dataset.

TABLE 4.3
Comparison results on NTU RGB+D Dataset

	cross-subject(%)	cross-view (%)
Lie Group, 2014 [28]	50.10	52.80
Skeletal Quads [8]	38.60	41.40
LieNet, 2017 [12]	61.30	67.00
HBRNN, 2015 [7]	59.10	64.00
Part-aware LSTM, 2016 [23]	62.90	70.30
ST-LSTM + Trust Gate, 2016 [17]	69.20	77.70
JTM, 2016 [34]	73.40	75.20
Ensemble LSTM, 2017 [15]	74.60	81.25
STA-LSTM,2017 [25]	73.40	81.20
GCA-LSTM, 2017 [18]	74.40	82.80
Two-stream RNN, 2017 [30]	71.30	79.50
JDM, 2017 [16]	76.20	82.30
CNN+LSTM, 2018 [21]	67.50	76.21
Multi-task RNN, 2018 [35]	-	82.60
Multiview Re-Observation Fusion, 2018 [9]	73.80	85.90
Beyond Joints, 2018 [31]	79.50	87.60
Our method (STdNet)	**82.55**	**88.03**

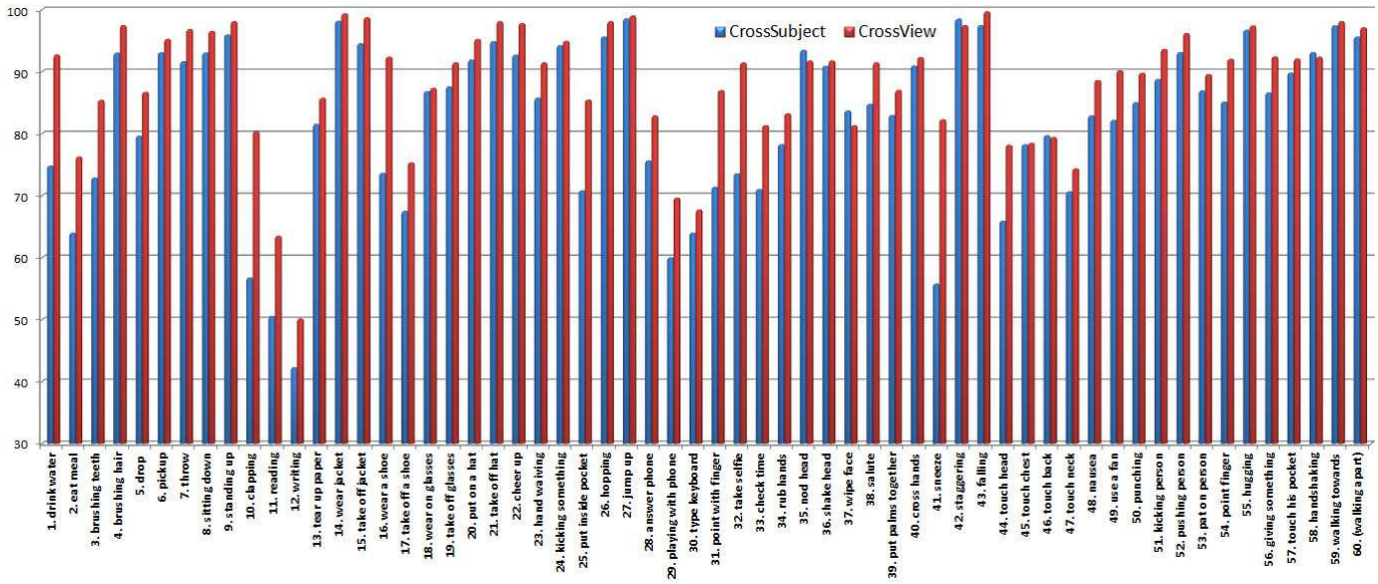


FIG. 4.1. *Recognition accuracies of each action class of NTU RGB+D dataset. There are 60 action classes in this dataset.*

5. Conclusion. This paper proposed an ensemble network consists of convolutional neural networks (CNN) and long short term memory (LSTM) neural networks. A CNN based subnet ($SdNet$) is designed to capture spatial information, where as LSTM based subnet ($TdNet$) exploits temporal dynamics along the video sequence. With the motivation of ensemble learning, these two subnets are fused as one ensemble network ($STdNet$). The efficacy of the proposed method is evaluated on two widely used datasets: UTD MHAD and NTU RGB+D datasets. Our $STdNet$ achieved competitive results with the recent works for action recognition task.

TABLE 4.4
Action classes pertaining to specific recognition range on NTU RGB+D dataset

Recognition range	cross-subject	cross-view
>=95%	**11** classes (9, 14, 15, 21, 26, 27, 42, 43, 55, 59, 60)	**19** classes (4, 6, 7, 8, 9, 14, 15, 20, 21, 22, 24, 26, 27, 42, 43, 52, 55, 59, 60)
90% - 94%	**13** classes (4, 6, 7, 8, 20, 22, 24, 35, 36, 40, 52, 57, 58)	**17** classes (1, 16, 19, 23, 32, 35, 36, 38, 40, 49, 50, 51, 53, 54, 56, 57, 58)
85% - 89%	**9** classes (18, 19, 23, 38, 50, 51, 53, 54, 56)	**8** classes (3, 5, 13, 18, 25, 31, 39, 48)
80% - 84%	**7** classes (5, 13, 37, 39, 46, 48, 49)	**6** classes (10, 28, 33, 34, 37, 41)
75% - 79%	**4** classes (1, 28, 34, 45)	**5** classes (2, 17, 44, 45, 46)
70% - 74%	**7** classes (3, 16, 25, 31, 32, 33, 47)	- **2** classes (47, 29)
65% - 69%	**2** classes (17, 44)	**2** classes (11, 30)
60% - 64%	**3** classes (2, 29, 30)	-
<=59%	**4** classes (10, 11, 12, 41)	**1** classes(12)

6. Acknowledgments. The authors would like to acknowledge the NVIDIA Corporation for the donation of Titan XP GPU under NVIDIA GPU Grant Program.

REFERENCES

[1] Z. Cai, L. Wang, X. Peng, and Y. Qiao, *Multi-view super vector for action recognition*, in Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2014, pp. 596–603.
[2] R. Chaudhry, F. Ofli, G. Kurillo, R. Bajcsy, and R. Vidal, *Bio-inspired dynamic 3d discriminative skeletal features for human action recognition*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2013, pp. 471–478.
[3] C. Chen, R. Jafari, and N. Kehtarnavaz, *Utd-mhad: A multimodal dataset for human action recognition utilizing a depth camera and a wearable inertial sensor*, in Image Processing (ICIP), 2015 IEEE International Conference on, IEEE, 2015, pp. 168–172.
[4] P. Climent-Pérez, A. A. Chaaraoui, J. R. Padilla-López, and F. Flórez-Revuelta, *Optimal joint selection for skeletal data from rgb-d devices using a genetic algorithm*, in Mexican International Conference on Artificial Intelligence, Springer, 2012, pp. 163–174.
[5] Z. Ding, P. Wang, P. O. Ogunbona, and W. Li, *Investigation of different skeleton features for cnn-based 3d action recognition*, in Multimedia & Expo Workshops (ICMEW), 2017 IEEE International Conference on, IEEE, 2017, pp. 617–622.
[6] Y. Du, Y. Fu, and L. Wang, *Skeleton based action recognition with convolutional neural network*, in Pattern Recognition (ACPR), 2015 3rd IAPR Asian Conference on, IEEE, 2015, pp. 579–583.
[7] Y. Du, W. Wang, and L. Wang, *Hierarchical recurrent neural network for skeleton based action recognition*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1110–1118.
[8] G. Evangelidis, G. Singh, and R. Horaud, *Skeletal quads: Human action recognition using joint quadruples*, in Pattern Recognition (ICPR), 2014 22nd International Conference on, IEEE, 2014, pp. 4513–4518.
[9] Z. Fan, X. Zhao, T. Lin, and H. Su, *Attention-based multiview re-observation fusion network for skeletal action recognition*, IEEE Transactions on Multimedia, 21 (2018), pp. 363–374.
[10] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
[11] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, Neural computation, 9 (1997), pp. 1735–1780.
[12] Z. Huang, C. Wan, T. Probst, and L. Van Gool, *Deep learning on lie groups for skeleton-based action recognition*, in Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE computer Society, 2017, pp. 1243–1252.
[13] M. E. Hussein, M. Torki, M. A. Gowayyed, and M. El-Saban, *Human action recognition using a temporal hierarchy of covariance descriptors on 3d joint locations.*, in IJCAI, vol. 13, 2013, pp. 2466–2472.
[14] T. Kerola, N. Inoue, and K. Shinoda, *Spectral graph skeletons for 3d action recognition*, in Asian Conference on Computer Vision, Springer, 2014, pp. 417–432.
[15] I. Lee, D. Kim, S. Kang, and S. Lee, *Ensemble deep learning for skeleton-based action recognition using temporal sliding lstm networks*, in 2017 IEEE International Conference on Computer Vision (ICCV), IEEE, 2017, pp. 1012–1020.
[16] C. Li, Y. Hou, P. Wang, and W. Li, *Joint distance maps based action recognition with convolutional neural networks*, IEEE Signal Processing Letters, 24 (2017), pp. 624–628.
[17] J. Liu, A. Shahroudy, D. Xu, and G. Wang, *Spatio-temporal lstm with trust gates for 3d human action recognition*, in

European Conference on Computer Vision, Springer, 2016, pp. 816–833.
[18] J. Liu, G. Wang, P. Hu, L.-Y. Duan, and A. C. Kot, *Global context-aware attention lstm networks for 3d action recognition*, in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 7, 2017, p. 43.
[19] M. Liu, H. Liu, and C. Chen, *Enhanced skeleton visualization for view invariant human action recognition*, Pattern Recognition, 68 (2017), pp. 346–362.
[20] M. Müller, T. Röder, and M. Clausen, *Efficient content-based retrieval of motion capture data*, in ACM Transactions on Graphics (ToG), vol. 24, ACM, 2005, pp. 677–685.
[21] J. C. Núñez, R. Cabido, J. J. Pantrigo, A. S. Montemayor, and J. F. Vélez, *Convolutional neural networks and long short-term memory for skeleton-based human activity and hand gesture recognition*, Pattern Recognition, 76 (2018), pp. 80–94.
[22] L. L. Presti, M. La Cascia, S. Sclaroff, and O. Camps, *Gesture modeling by hanklet-based hidden markov model*, in Asian Conference on Computer Vision, Springer, 2014, pp. 529–546.
[23] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang, *Ntu rgb+ d: A large scale dataset for 3d human activity analysis*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 1010–1019.
[24] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake, *Real-time human pose recognition in parts from single depth images*, in Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, Ieee, 2011, pp. 1297–1304.
[25] S. Song, C. Lan, J. Xing, W. Zeng, and J. Liu, *An end-to-end spatio-temporal attention model for human action recognition from skeleton data.*, in AAAI, vol. 1, 2017, pp. 4263–4270.
[26] S. Sun, *A survey of multi-view machine learning*, Neural Computing and Applications, 23 (2013), pp. 2031–2038.
[27] A. Taalimi, A. Rahimpour, C. Capdevila, Z. Zhang, and H. Qi, *Robust coupling in space of sparse codes for multi-view recognition*, in 2016 IEEE International Conference on Image Processing (ICIP), IEEE, 2016, pp. 3897–3901.
[28] R. Vemulapalli, F. Arrate, and R. Chellappa, *Human action recognition by representing 3d skeletons as points in a lie group*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 588–595.
[29] C. Wang, Y. Wang, and A. L. Yuille, *An approach to pose-based action recognition*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 915–922.
[30] H. Wang and L. Wang, *Modeling temporal dynamics and spatial configurations of actions using two-stream recurrent neural networks*, in e Conference on Computer Vision and Pa ern Recognition (CVPR), 2017.
[31] H. Wang and L. Wang, *Beyond joints: Learning representations from primitive geometries for skeleton-based action recognition and detection*, IEEE Transactions on Image Processing, 27 (2018), pp. 4382–4394.
[32] P. Wang, W. Li, Z. Gao, Y. Zhang, C. Tang, and P. Ogunbona, *Scene flow to action map: A new representation for rgb-d based action recognition with convolutional neural networks*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 595–604.
[33] P. Wang, W. Li, P. Ogunbona, J. Wan, and S. Escalera, *Rgb-d-based human motion recognition with deep learning: A survey*, Computer Vision and Image Understanding, 171 (2018), pp. 118–139.
[34] P. Wang, Z. Li, Y. Hou, and W. Li, *Action recognition based on joint trajectory maps using convolutional neural networks*, in Proceedings of the 2016 ACM on Multimedia Conference, ACM, 2016, pp. 102–106.
[35] H. Wangg and L. Wang, *Learning content and style: Joint action recognition and person identification from human skeletons*, Pattern Recognition, 81 (2018), pp. 23–35.
[36] S. Zhang, X. Liu, and J. Xiao, *On geometric features for skeleton-based action recognition using multilayer lstm networks*, in 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, 2017, pp. 148–157.
[37] L. Zhou, W. Li, Y. Zhang, P. Ogunbona, D. T. Nguyen, and H. Zhang, *Discriminative key pose extraction using extended lc-ksvd for action recognition*, in Digital lmage Computing: Techniques and Applications (DlCTA), 2014 International Conference on, IEEE, 2014, pp. 1–8.

Edited by: Shajulin Benedict
Received: March 31, 2019
Accepted: August 30, 2019

DOI 10.12694/scpe.v20i3.1423
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 3, pp. 495–509. http://www.scpe.org

VIRTUAL CHANNEL AWARE SCHEDULING FOR REAL TIME DATA-FLOWS ON NETWORK ON-CHIP

MOHAMMED AMINE MEGHABBER[*], LAKHDAR LOUKIL[†], RICHARD OLEJNIK[‡], ABOU EL HASSAN BENYAMINA[§], AND ABDELKADER AROUI[¶]

Abstract. The increasing complexity of real-time applications presents a challenge to researchers and software designers. The tasks of these applications usually exchange many data-flows and often need to satisfy real-time constraints. Although the Network on-Chip (NoC) paradigm offers an underlying communication infrastructure that gives more hardware resources, tasks and data-flows cannot be performed before their deadlines. In recent works, preemptive wormhole switching with fixed priority has been introduced to meet real-time constraints of real-time applications. However, it suffers some bottleneck such as hardware requirement where none of these works takes account of the number of implemented Virtual Channels on the router. To alleviate this problem, we propose a novel scheduler for soft real-time data-flows application that takes into account the lack on resource in routers in term of Virtual Channels. Experimental results obtained on a benchmark of synthetic and soft real applications have shown the efficiency of our approach in term of real-time constraints satisfaction for data-flow traffics and hardware requirements.

Key words: Embedded Systems, Network on-Chip, Real Time systems, Router, Wormhole.

AMS subject classifications. 68M10, 68M14, 68M20

1. Introduction.

As the technology advances, the number of processors in a chip grows [1] due to the transistor integration progress. In the last three decades, the hardware (processors) evolution was exponential and the Moore law [2] which states that "the number of transistors on an integrated circuit doubled every year", has reached its physical limits due to heat dissipation and energy consumption that increased with the number of processors. This led nano-architecture engineers to propose a new system-on-chip (SoC) architecture called network-on-chip (NoC) [3].

Inspired from the classical network, Networks on-Chip (Fig. 1.1) have emerged as a new communication paradigm in systems on chip [4]. Unlike classical Multiprocessors System on-Chip (MPSoC) shared bus architectures [5] which do not allow scaling and behave unpredictable when connecting up to 10 processors [6], NoC systems offer high scalability and improved parallelism [4]. Currently, many systems on-chip manufacturers have developed commercial NoCs such as TeraFLOPS on-chip network developed by Intel [7] and TILEPro64 developed by Tilera [8].

On the other hand, embedded applications such as those found in aerospace domain [9] are increasingly complex and subject to strong energy consumption and real-time constraints that require efficient and high performance execution frameworks. Indeed, tasks of such applications are interdependent and usually exchange many messages during their execution on processing elements (PE), which requires an efficient underlying communication infrastructure like the one provided by a NoC. Furthermore, these dataflows are mostly subject to soft or hard real-time constraints on their period and their deadline, for example. In this conditions, dataflows must arrive at their destination nodes (Processing Element) in time and any exceeding deadlines can at best degrade the application performance and at worst have desastrous consequences.

Due to its small buffering requirement, high throughput and low latency, the wormhole switching technique [10] is the most common flow control mechanism in NoC routers (Fig. 1.2). In this switching technique, a dataflow is split into packets, and a packet is split into flits[1] (Fig. 1.3), where header flit embeds routing information and data flits contain user data. When a router receives a flit, it is first buffered in an input Virtual Channel (VC) [11] and then the arbiter function of the current router determines which of the candidate flits the one that will be routed to the next router. One of the most commonly used arbitration technique is round robin (RR) [12]. RR handles VC in circular order and ignores packet characteristics such as priority of a packet, which

[*]Computer Science Department, University of Oran 1 Ahmed Ben Bella, Algeria (meghabber.mohammed@edu.univ-oran1.dz).
[†]Computer Science Department, University of Oran 1 Ahmed Ben Bella, Algeria. (loukil.lakhdar@univ-oran1.dz).
[‡]CNRS, Univ. Lille, Centrale Lille, IMT Lille Douai, Lille, France (richard.olejnik@univ-lille1.fr).
[§]Computer Science Department, University of Oran 1 Ahmed Ben Bella, Algeria (abouelhassan.benyamina@univ-oran1.dz).
[¶]Satellite development Center, Algeria (aaroui@cds.asal.dz).
[1]FLow control unIT. The data unit processed by a router.

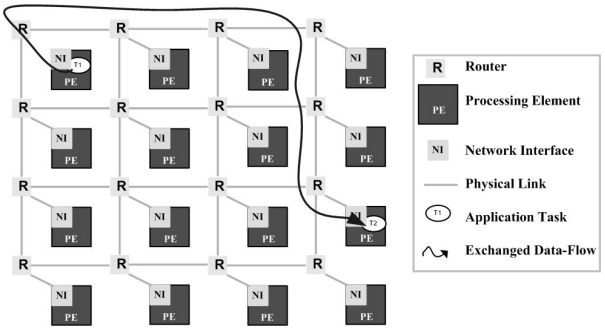


FIG. 1.1. *2D mesh network-on-chip.*
The figure shows 2D-mesh Network on chip topology where a set of processors are interconnected via routers. NI formats exchanged messages between application tasks to packets then flits.

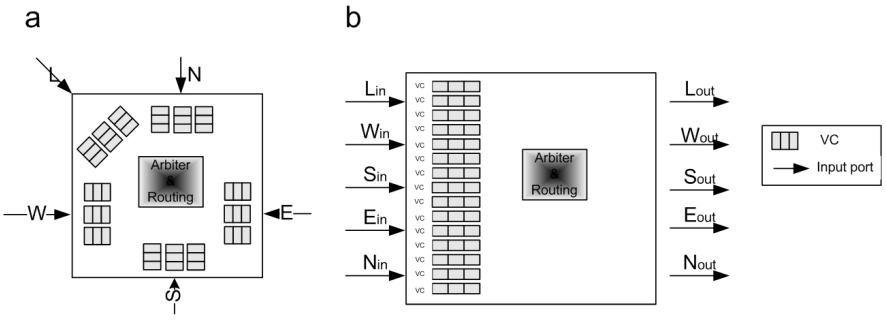


FIG. 1.2. *Interface views of a typical router: (a) Architectural view; (b) Functional view.*
The figure shows the router architecture. It includes five input and output port, arbiter and routing function. Each Input physical port includes a set of Virtual Channels (VC) connected to buffers. Each buffer has a limited amount of space for flits. For each cycle, the arbiter and routing fabric select one flit to be routed through the output port. The output port is a physical wire that transfers flits to next router.

makes it unsuitable for real-time applications where dataflows have priorities and must be routed according to their priority. To deal with prioritized dataflows, a wormhole switching with a fixed priority preemption has been proposed [13]. In this switching technique, Virtual Channels of an input port are labeled with a priority value and each new flit arriving at a router is assigned to the Virtual Channel labeled with the priority of the flit. This implies that the number of Virtual Channels per input port should be at least equal to the number of distinctive flit priorities. This constraint is not always feasible. On the one hand, nowaday real-time applications are more and more greedy and, on the other hand, it is technically not feasible to manufacture routers with a large number of VC. We notice, for instance, that the experimental Intel Single-chip Cloud Computer (SCC) NoC [7] implements 8 VC per input port.

In this work, we propose a general-purpose Virtual Channel allocation technique for wormhole switching with priority pre-emption which is able to schedule soft real-time applications by assigning a Virtual Channel to a flit of a given priority whatever the complexity of the application (number of priorities) and whatever the architecture of the router (number of VC per input port). This paper is structured as follows. Section 2 presents recent works related to real-time dataflows scheduling. Section 3 describes the application model targeted in this work. Section 4 depicts the VC allocation technique we propose. Then, We report and analyse simulation results performed on synthetic and real-life applications (Sect.5). Finally, we conclude and give some

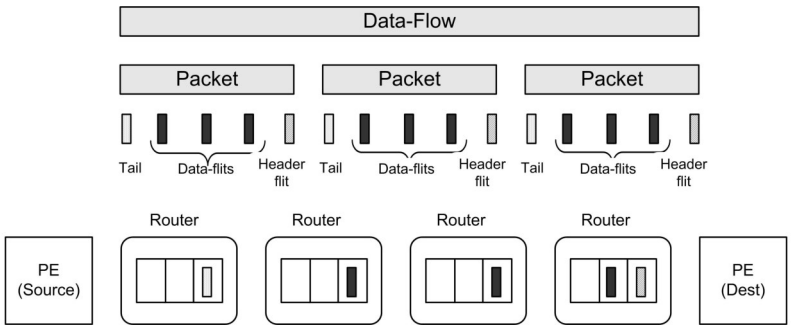


FIG. 1.3. *Packet formatting in wormhole switching.*

An exchanged message is split into packets, and a packet is split into flits. The flit represents the data unit processed by the router. During packet transmission, flits of the packet can be stored in multiple routers (buffers). If a contention occurs, just a flit will be blocked not the entire packet.

perspectives.

2. Related Work. During the last few years, real-time dataflows scheduling problem on NoC-based systems on chip has been widely studied. Various scheduling methods have been proposed in the literature where the objective is to achieve the real-time constraints of embedded applications using a minimum hardware resources. In this section, we will report the most important work related to this problem.

The preemptive wormhole switching with fixed priority was firstly proposed by Balakrishnan et al. [14] to address the problem of real-time communication in multiprocessor networks. This technique has subsequently been adopted by several researchers to estimate the worst case response time for wormhole switching. Hary et al. [15], in their paper, consider the links traversed by a dataflow as a single shared path and propose their analysis accordingly, while Kim et al. [16] attempt to construct a dependency graph for its response time analysis.

Shi et al. [17] have adapted the response time analysis in a fixed priority processor proposed in [18] for the analysis of the response time in the case of several dataflows. The authors consider that the NoC has as many Virtual Channels (VC) as dataflows, that the VC and dataflows have priorities and that any given priority dataflow is assigned to the VC that has the same priority. Such proposal presents scaling problems since it is technically unfeasible to have routers with a large number of VC. The implementation of a VC is indeed expensive in terms of manufacturing cost and logic area. Mello in [19] has stated that while the Hermes NoC with a single VC took 17% of the logic area of their hardware test-bed, the design with two and four VC took 32.61% and 75.41% of the logic area respectively. We notice that input ports of routers of a Intel TeraFLOPS NoC [7] developed by Intel has 8 Virtual Channels.

In order to improve their proposal and reduce hardware resources required to implement their scheduling strategy, She et al. in [20] suggest another scheduling algorithm that partitions dataflows into clusters and assigns a unique priority to dataflows of a cluster. This allows to reduce the number of Virtual Channels (since dataflows of a cluster will be assigned to a single Virtual Channel) and achieve real-time constraints but may give rise to unpredictable network latency and blocking problem. The latter can take place since a VC stores different packets shared the same priority and served in FIFO because the priority preemption is only available between VC and therefore a packet can be blocked by another packet with the same priority which holds a VC which can in turn block other packets, and so on.

In [21], authors propose a Dynamically Changing Virtual Channels (DCVC) scheduling technique where each packet still maintains its priority constant during an entire traversal of the NoC but may occupy different Virtual Channels within routers on its path. According to the authors, this considerably reduces the number of VC. DCVC however requires the header flits to hold the list of all Virtual Channels of the traversed path which will overload the routers and induce scalability issues. The same authors suggest in [22] a dynamic priority policy EDF (Earliest Deadline First) as arbiter in such a way to improve schedulability. The EDF scheduler allows jobs (task instance) to change their priority dynamically on runtime according to their deadline. A worst case time analysis has been performed but no implementation has been done. In their last work [23], Nikoli at

al. propose a timing analysis for arbitrary VC size where they prove that bigger Virtual Channel buffers do not necessarily help tasks and data-flows to be performed before their deadlines.

Sudev et al. [24] who consider that a preemptive router is expensive in terms of implementation costs and energy consumption, adopt a real-time non-preemptive router with fixed priority. To process high priority dataflows, they propose a technique named PTF (Priority Forwarding and Tunneling) which consists in temporarily increase the priority of low-priority packets that prevent the timely transmission of high-priority packets and thus avoid the head of line (HoL) phenomenon.

In a later work [25], the authors propose an arbitration technique for non-preemptive routers. In their proposal, the router suspends the transmission of the current packet upon the arrival of a packet of higher priority. In that case, the router split the transmission by sending a tail flit followed by the construction of a new header that initiates a new arbitration request on that router. This would allow the higher priority packet to start transmission. Like that, they emulate the preemption.

Authors in [37] consider proposed methods in the literature as optimist, and they have improved their analysis through the distinction between downstream and upstream indirect interferences. However, they do not take into account the number of available VC. Afterwards, Giroudot at al. [38], have extended [37] to support the backpressure and flow serialization.

All works presented above propose solutions to schedule real-time data-flows and aim to achieve real time constraints. However, they all consider that the NoC has as many VC as dataflows. In practice, it is unfeasible to have routers with a large number of VC. Our challenge is to propose a lightweight algorithm which takes into account the hardware limitation in term of VC and schedules dataflows whatever the complexity of the application (number of priorities) and whatever the architecture of the router (number of VC per input port). In this paper, we propose a new Virtual Channel allocation and assignment technique, which reduces significantly the number of needed VC implemented in router. Otherwise, each dataflow could find a VC to be allocated even where there is not as many VC as dataflows. This technique relaxes hardware requirements in term of buffers (VC) and improve the real-time application schedulability.

3. Application Model. An application is modeled by a finite set $\Omega = \{\delta_1, \delta_2, \cdots, \delta_n\}$ of n real-time dataflows δ_i. Each dataflow δ_i is characterized by the following parameters:

- S_i: Source node,
- $Dest_i$: Destination node,
- A_i: Arrival time
- L_i: Basic latency which is the maximum transmission time from the source node S_i to the destination node $Dest_i$. This parameter is estimated assuming no congestion in the network,
- p_i: Period (elapsed time between successive releases of packets of δ_i),
- D_i: Deadline which is the due time that the scheduling of the dataflow must not exceed,
- P_i: Priority.

The basic latency L_i can be evaluated using the following formula [26]:

$$L_i = H \times d_R + F \times d_T, \tag{3.1}$$

where :

- H is the number of hops from the source processing element to destination processing element.
- d_R is the time it takes for a router to switch a flit.
- d_T is the transfer time of a flit from a router to a neighboring router.
- F is the number of data flits in a packet. F is given by the following equation:

$$F = \frac{size(Packet) + A_d}{size(Flit)}, \tag{3.2}$$

where A_d is the size of the additional data (header and tail flits).

For a regular 2D-mesh NoC, the number of hops H is equal to :

$$H = |R_d^x - R_s^x| + |R_d^y - R_s^y|, \tag{3.3}$$

where:

- R_s^x and R_s^y are respectively the x and y coordinates of the source processing element,
- R_d^x and R_d^y are respectively the x and y coordinates of the destination processing element.

Hereafter, we will assume that d_R=d_T=1. So, Eq. 3.1 becomes as follows:

$$L_i = |R_d^x - R_s^x| + |R_d^y - R_s^y| + F. \tag{3.4}$$

4. Modulo-based VC Allocation and Assignment. As mentioned in Sect.2, most of the work dealing with scheduling real-time applications in NoC consider routers running a fixed priority preemptive wormhole switching to schedule real-time dataflows [16, 15, 27, 22]. In such routers, a newly arrived high priority header flit preempts a low priority flit being transmitted. Authors of these works generally assume that there are at least as many available VC per input port as priorities. Such assumption is technically unrealistic since the number of VC per input port in real-life NoCs is usually much more smaller than the number of priorities of real-time dataflows. For instance, Intel TeraFLOPS NoC [7] implements 8 buffers per input port. Moreover, hardware experience results [19, 28] show that implementing additional buffers is an expensive process.

In this work, we propose a general-purpose VC allocation and assignment strategy named *Modulo-based VC allocation and assignment technique* (designated hereafter by MVCAA). The qualifier "general-purpose" is used to say that MVCAA allows scheduling soft real-time dataflows whatever the number of VC in a router and whatever the size of real-time applications (the number of priorities).

Algorithm 1 depicts the pseudo-code of MVCAA. Overall, MVCAA proceeds in two steps. The first step (Sect.4.1) is the VC allocation and assignment that determines the VC that will host flits of each incoming dataflow δ_i of priority P_i. The second one (Sect.4.2) is the scheduling step that determines how the arbiter selects the VC to process.

Algorithm 1 Pseudo-code of modulo-based VC allocation and assignment (MVCAA).

```
Input: AS= {(δ1,P1), (δ2,P2), ···, (δn,Pn)} of n couples (Flit, Priority)
       VC={VC1, VC2, ···, VCm } of m Virtual Channels, where m << n
Output: ALOC= {(δ1, VCi1), (δ2,VCi2), ···, (δn,VCim)} of couples (Flit, Allocated Virtual Channel)
for (Arriving flit δi) do
    if ( δi is a header flit) then
        BufferTag = Pi%|VC|;
//Allocate VC_BufferTag and assign δi to VC_BufferTag;
        Allocate(δi, VC_BufferTag)
//Update the VC_BufferTag priority register;
        UpdatePriorityRegister(VC_BufferTag, Pi);
    else//δi is a data flit, in which case assign it to VC_BufferTag;
        Assign(δi, VC_BufferTag);
    end if
end for
```

4.1. Step 1: Assigning a VC to a dataflow. The architecture of a router considered in the present work is outlined in Fig.4.1. Each input port has a number of Virtual Channels and each Virtual Channel has a priority tag register (represented by a square on the VC) indicating the current priority of the VC (i. e. priority of flits the VC currently hosts). Thereby for each incoming dataflow δ_i of priority P_i, the Virtual Channel VC_j that would be allocated to that dataflow is determined using the following equation:

$$\forall \delta_i \in \Omega, \delta_i \text{ is assigned to } VC_j \text{ where } j = P_i\%(|VC|), \tag{4.1}$$

where:

- VC: the set of Virtual Channels of a physical input port of a router.
- Ω: the set of dataflows of a real-time application.

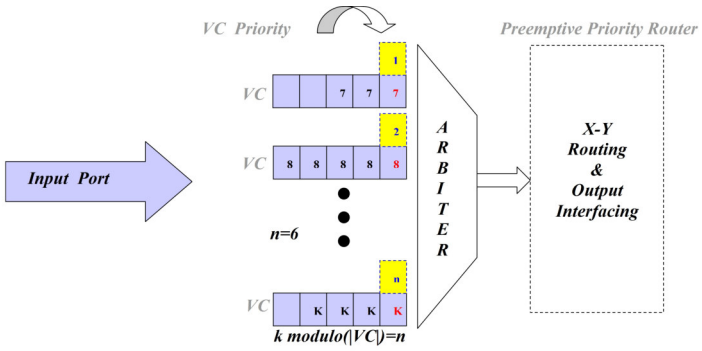


FIG. 4.1. *Dynamic VC priority for wormhole Router*

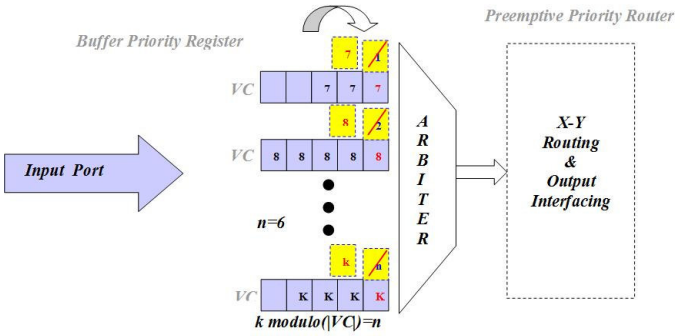


FIG. 4.2. *Updating VC tags.*

If VC_j is free, it is locked and allocated to the dataflow δ_i, and its priority tag register is set to the priority P_i of the dataflow δ_i (Fig. 4.2). The lock of VC_j will be released by the tail flit of the packet. Otherwise, the current dataflow δ_i is ignored and the next incoming dataflow is considered.

4.2. Step 2: Scheduling dataflows. At each cycle, the router assigns an output port to each incoming packet (stored in an input VC) according to its destination using a flow priority-based adaptive routing (see Sect.4.3). To this purpose, we implement a priority-based arbiter that scans input VC of the router input port to look for the highest priority packet. If the highest priority packet cannot be routed to the adjacent router for lack of buffer space in that router, the next highest priority packet is considered. To be aware of the buffers status of adjacent routers, a credit-based flow control [29] is performed. In credit-based flow control, a *credit counter* register associated to each output port indicates the credits of input buffers of the adjacent router in term of free slots. It sends a ready signal to the arbiter when the number of available buffer slots at the output buffer is greater than zero. When a header flit is selected to be routed to the next router, the input buffer of the adjacent router buffer is be locked. This lock will be released once the tail flit leaves the buffer.

4.3. Adaptive XY routing. XY routing is the most popular routing algorithm for 2D mesh or torus NoC topologies. It first routes a packet in X (horizontal) direction then in Y (vertical) direction. The asset of XY routing is that it never runs into deadlock or livelock [30]. Its disadvantage is that it produces a high load in the center of the network which can cause a hot spot.

For a better load balancing, we implemented a *half adaptive XY routing* [30] where the router first tries to route the flit in X direction and assigns a free VC if it exists. Otherwise, the router routes the flit in Y direction. If no VC is free in the two directions, the flit is blocked and the router serves the next dataflow. The pseudo-code of Priority based half adaptive XY routing is depicted in Alg.2.

Algorithm 2 Priority based half adaptive XY routing.

```
1: Input: Flit δ_i of priority P_i, source node S_i and destination node Dest_i
2: Ouput: nextRouter, destVC
3: destVC = P_i%|VC|;
4: nextRouter = xyRouting(S_i, Dest_i);
5: if (vcAvailable(nextRouter, destVC)) then
6:     lock(nextRouter, destVC);
7:     moveFlit(δ_i, nextRouter, destVC);
8: else
9:     nextRouter = yxRouting(S_i, Dest_i);
10:    if vcAvailable(nextRouter, destVC) then
11:        lock(nextRouter, destVC);
12:        moveFlit(δ_i, nextRouter, destVC);
13:    else
14:        nextRouter = -1;
15:        destVC = -1;
16:    end if
17: end if
```

5. Experimental results and analysis. In this section, we will report some experimental results carried out in this study. The objective of these experiments is twofold: the first is to test the performance of our proposal (i.e. MVCAA technique, see Sect.4) by comparing it with some state-of-the-art real-time scheduling methods for NoC-based mutiprocessor system-on chip, the second is to test the scalability of MVCAA when increasing the size of the NoC. The performance metric retained is the ratio of the number of dataflows that are missing their deadlines on the number of real-time dataflows injected into the NoC. We recall that a dataflow δ_i misses its deadline if its response time R_i is greater than its deadline D_i. The response time is given by the following expression:

$$R_i = L_i + B_i, \tag{5.1}$$

where L_i is the basic latency (see Sect.3) and B_i is the time during which the dataflow δ_i is blocked in a buffer by higher priority dataflows. In our case, an estimate of R_i is provided by the NoC simulator [31] and corresponds to the latency when the dataflow δ_i reaches its PE destination.

We have compared MVCAA with two state-of-the-art real-time scheduling methods for NoC-based mutiprocessor system-on chip: Priority Share Policy (PSP) of Shi and al. [20], and Dynamically Changing Virtual Channels (DCVC) of Nikolič and al. [21] where both aim to reduce the number of Virtual Channels for priority-preemptive NoCs.

Experiments have been carried out on the NoC in-house simulator published in [31]. The simulator implements a 2D-mesh NoC topology and a wormhole switching technique, uses adaptive XY routing (Alg.2) and the credit-based flow control. It allows the user to keep track in real-time of the communications and highlights routers that suffer from congestion. The input of the simulator is an `XML` file that describes the application dataflows and the NoC configuration. An application is a set of dataflows and each dartaflow is characterized by a deadline, a period, an arrival time, a basic latency, a type (header or data flit) and a priority. The NoC is described by its topology (2D-mesh), the arbitration technique (MCVAA, PSP, DCVC), the NoC size (number of routers in each dimension), the number of VC per input port and the VC size (number of slots per VC). Algorithm 3 depicts the simulator kernel. The simulation output is an `XML` file that reports, for each simulation cycle, buffer states of the different routers, traffics, latency and missed deadlines of different packets over the entire NoC. For each cycle and each router and according to the arbitration technique, a dataflow is identified and routed to the adjacent router using half adaptive XY routing.

Two categories of applications have been considered: synthetic and real-life applications. Synthetic real-time applications have been generated using the random generator proposed in [32]. Three real-life real-time

Algorithm 3 NoC simulator kernel [31].

```
Input: AS= {(δ_i, P_i)/ δ_i ∈ Ω, P_i: priority of dataflow δ_i};
       NoC architecture.
Output: NI_ALOC={(δ_i,NI_K)/ NI_K: Network Interface}
load(Application)
load(NoC architecture)
for cycle = 0 to numCycles do
    for each NI in NoC do
        if (NI has traffic to forward) then
            moveTrafficFromNIToLocalRouter ;
        end if
    end for
    for Each Router R in NoC do
        for Each Input port of R do //Define output port for each incoming flit;
            UpdateRouterRequests(Routing);
        end for
        for Each Output port of R do
            CheckCandidateFlitsPerOutputPort;
            RunArbitrationToSelectGrantedFlit;
            SwitchGrantedFlitToNextRouter;
            MVCAA(AS, VC, ALOC);
    end for
    end for
end for
```

TABLE 5.1
Experimental protocol.

NoC sizes	8×8 and 16×16 2D-mesh
Buffering	4, 5, and 8 VC per input port
Switching	Preemptive wormhole
Arbiter	Per priority
Flow control	Credit-based
Routing	Half Adaptive XY routing
Application sizes	10, 20, 30, 40, 50, 70, 80, 100 dataflows

applications have been selected: Video Object Plane Decoder (VOPD) [33, 34], MPEG4 [35], and an autonomous vehicle application (AutoV) [36]. Table 5.1 summarizes the experimental protocol adopted in our experiments.

5.1. Synthetic applications. Figures 5.1-5.6 give the percentage of missed deadlines for application sizes ranging from 10 to 100 dataflows and for VC numbers equal to 4, 5 and 8. 100 executions were performed for each configuration and the average value was considered. Figures 5.1-5.3 (8×8 2D mesh NoC) show that, for all configurations, `MVCAA` clearly outperforms `DCVC` and behaves almost like `PSP` in terms of the percentage of missed deadlines. However, for 16×16 2D mesh NoC, `MVCAA` outperforms both `PSP` and `DCVC` (see Figs.5.4-5.6) and the gap grows with the increase of the size of the application. This proves the robustness of our approach to increasing the size of the NoC. The performance degradation of `PSP` and `DCVC` can be explained by the fact that for `PSP` the packet blocking duration (see section 2) increases with the size of NoC (in term of the number of hops) and will have a negative impact on the number of missed dealines. Regarding `DCVC`, increasing the size of the NoC induces the increase of the number of hops and then of the size of `DCVC` packets which consequently increases the latency of the packet.

To study the impact of NoC architecture (Number of routers, number of VC per input port) on MVCAA schedulability of real-time dataflows, we considered 3 NoC sizes (7×7, 8×8 and 16×16) each with diferent

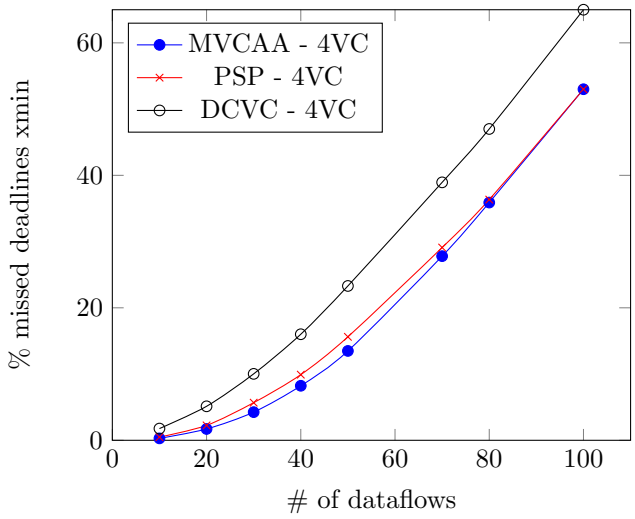


FIG. 5.1. *Percentage of missed deadlines on a 8×8 2D mesh NoC (4 VC).*

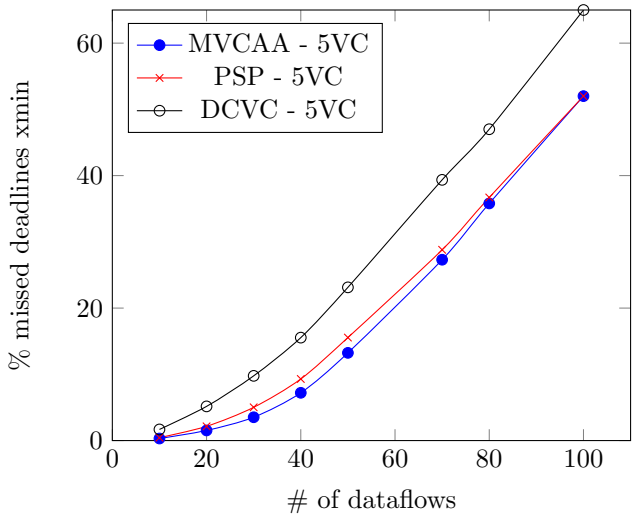


FIG. 5.2. *Percentage of missed deadlines on a 8×8 2D mesh NoC (5 VC).*

numbers of VC per input port (4, 5 and 8). Experimental results are illustrated by Figs. 5.7, 5.8 and 5.9. Several remarks can be drawn from these figures. The first one is that the increase in the number of VC while maintaining the architecture unchanged does not significantly impact the schedulability of real-time applications. This corroborates the fact that multiplying the number of VC in a router does not significantly improve the scheduling of dataflows, contrary to what is stated in some works as in [16, 17]. Consider the fact that the router always serves the highest priority packet, it seems that additional VC do not increase the application performance since all lowers priority packets have to wait for the transmission of the higher one. The other remark that we can extricate is that by slightly increasing the size of the NoC, one can gain much in quality of scheduling. We notice indeed that the 16×16 2D-mesh NoC reduces by more than 50% the number missed deadlines compared to the 8×8 NoC.

5.2. Real-life application. Three real-time embedded benchmarks are considered to evaluate the performance of the proposed `MVCAA`: Video Object Plane Decoder (`VOPD`) application [33, 34], `MPEG4` application [35]

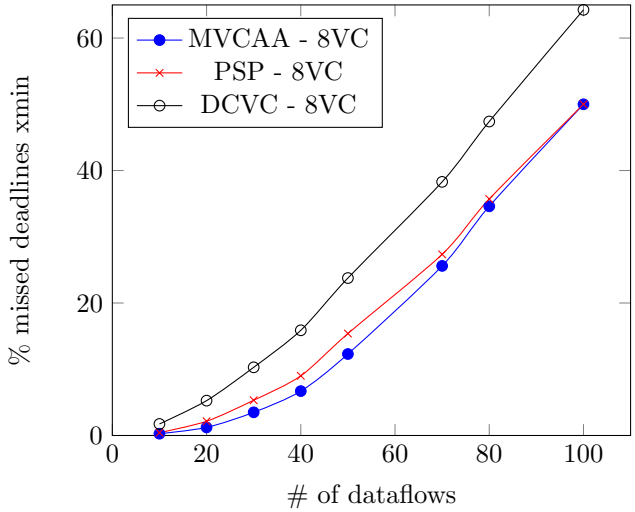


FIG. 5.3. *Percentage of missed deadlines on a 8×8 2D mesh NoC (8 VC).*

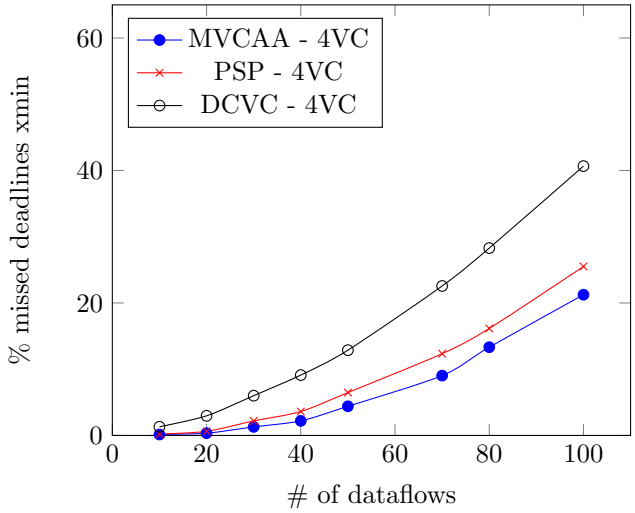


FIG. 5.4. *Percentage of missed deadlines on a 16×16 2D mesh NoC (4 VC).*

and an autonomous vehicle (`AutoV`) application [36].

`VOPD` application consists of 17 tasks that exchange data. Figure 5.10 depicts the application architecture, circles represent tasks and arcs represent exchanged data volume. Values on arcs represent the volume of data exchanged between tasks connected by the arc. Each task is characterized by three parameters: task period, capacity and deadline.

`MPEG4` consists of 20 tasks and 23 exchanged dataflows. Figure 5.11 gives the application task graph. Each task is characterized by its task period, its capacity and its deadline, the value on an arc represents the volume of data exchanged between tasks connected by the arc.

Autonomous vehicle application (`AutoV`) is characterized by large volume of dataflows (some dataflows may contain more than 38000 flits) and control loops with short messages. `AutoV` consists of 33 tasks and 38 dataflows.

We execute these applications on a 8×8 2D-mesh NoC having 8 VC per input port on different dataflow period instances. The results of the experiments illustrated in Fig. 5.12 show that `MVCAA` clearly outperforms

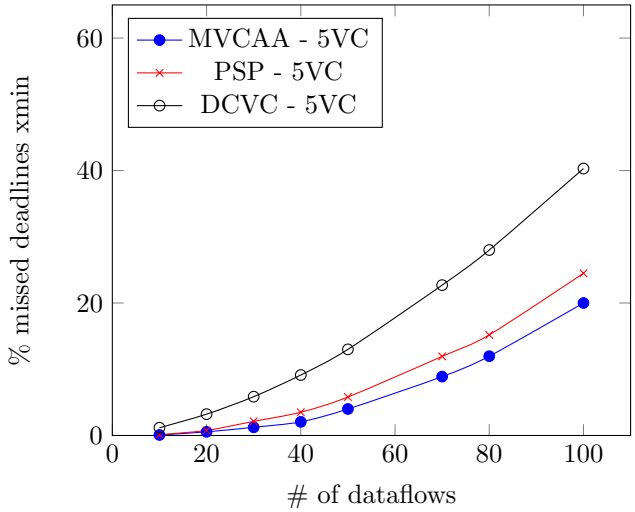


FIG. 5.5. *Percentage of missed deadlines on a 16×16 2D mesh NoC (5 VC).*

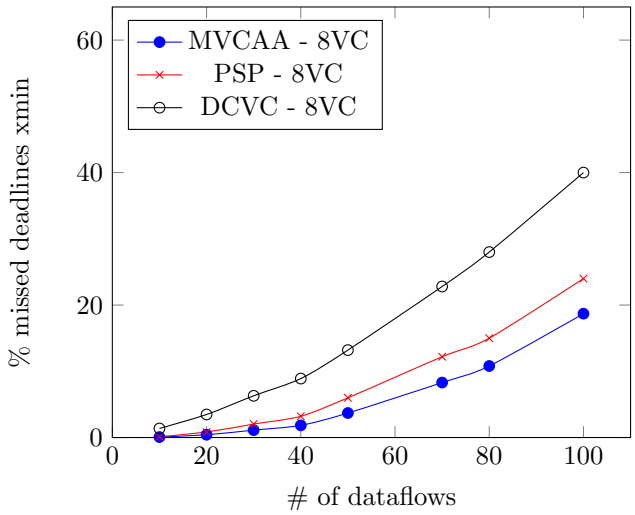


FIG. 5.6. *Percentage of missed deadlines on a 16×16 2D mesh NoC (8 VC).*

PSP and DCVC in terms of the percentage of dataflows that have saved their deadlines. The experimental results show that MVCAA achieves up to 19% for VOPD, 10% for MPEG4. For AutoV application, the improvement is 5% since the application exchanges huge and voluminous dataflows .

6. Conclusions. Networks-on-chip are an emerging technology and a promising communication paradigm offering more scalability and parallelism. Priority-based arbitration techniques have been proposed to schedule real-time applications. These techniques assume, however, that there are at least as many VC per router input port as there are dataflows. In this work, we have proposed a new priority-based arbitration technique named *Modulo-based VC Allocation and Assignment (MVCAA)* that takes into account the number of available VC in NoC and allows scheduling real-time dataflows regardless of the number of priorities and VC. Experiments have shown that MVCAA outperforms some state-of-the-art real-time scheduling methods.

As future works, we think that exploring new routing techniques, mapping and priority assignment algorithms would be more promising for real-time applications than implementing more VC within a router.

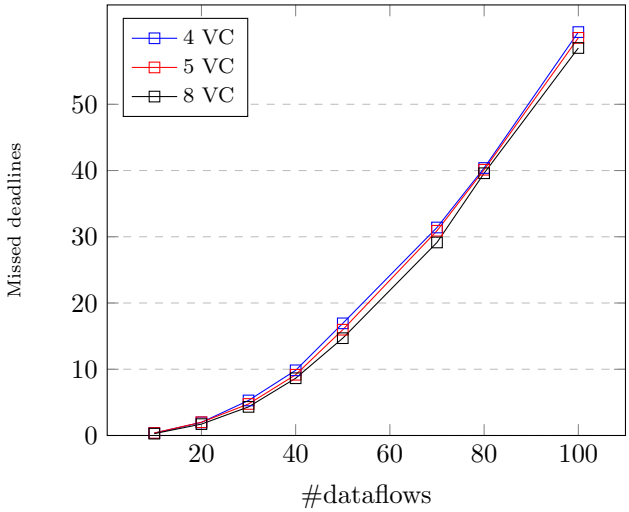


FIG. 5.7. *7×7 NoC.*

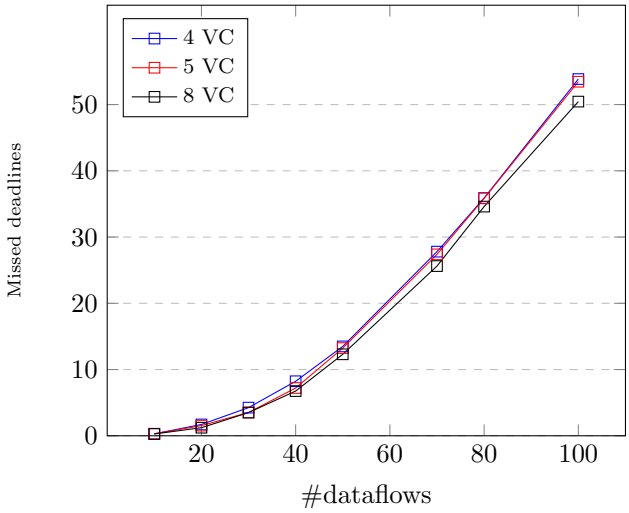


FIG. 5.8. *8×8 NoC.*

Acknowledgments. This work has been supported in part by the PHC TASSILI 14MDU917 project of the University of Oran 1, and by IRCICA,Université de Lille, USR 3380, F-59650 Villeneuve d'Ascq, France.

REFERENCES

[1] A. JERRAYA, H. TENHUNEN, AND W. WOLF. *Guest Editors' Introduction: Multiprocessor Systems-on-Chips.* Computer, 38(7):36–40, July 2005.

[2] G. E. MOORE. *Cramming more components onto integrated circuits.* In Mark D. Hill, Norman P. Jouppi, and Gurindar S. Sohi, editors, Readings in Computer Architecture, pages 56–59. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

[3] L. BENINI AND G. DE MICHELI. *Networks on chips: a new SoC paradigm.* Computer, 35(1):70–78, January 2002.

[4] T. BJERREGAARD AND J. SPARSO. *Implementation of guaranteed services in the MANGO clockless network-on-chip.* IEE Proceedings - Computers and Digital Techniques, 153(4):217–229, July 2006.

[5] S. FURBER. *Future trends in SoC interconnect.* In 2005 IEEE VLSI-TSA International Symposium on VLSI Design, Automation and Test, 2005. (VLSI-TSA-DAT)., pages 295–298, Hsinchu, Taiwan, Taiwan, April 2005. IEEE.

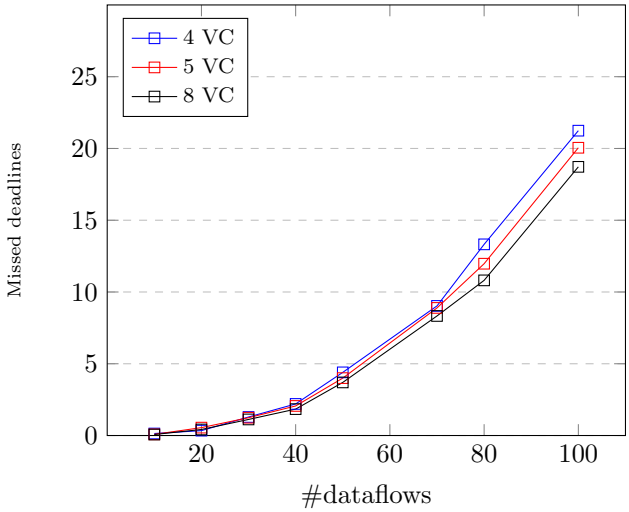


FIG. 5.9. *16×16 NoC.*

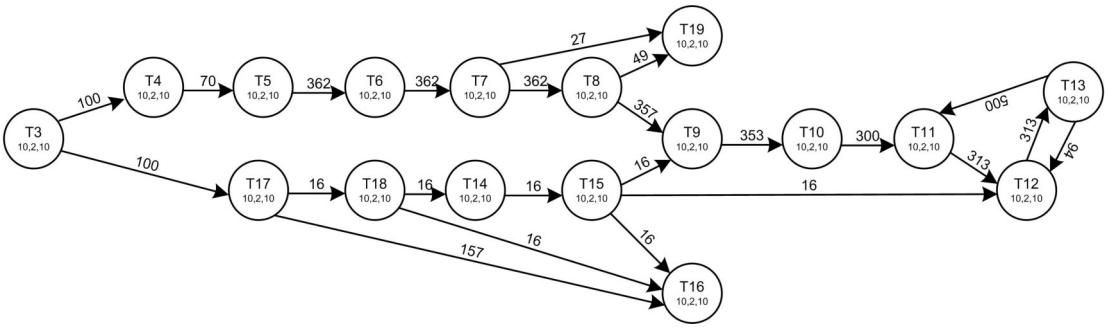


FIG. 5.10. *VOPD tasks graph.*

[6] JANTSCH. *Networks on Chip.* Springer US, New York, NY, USA, 2004.
[7] INTEL. *Single-Chip Cloud Computer (SCC),* 2010.
[8] CORPORATION TILERA. *Tile Processor Architecture Overview for the TILEPro Series,* 2013.
[9] F. BONIOL. *New Challenges for Future Avionic Architectures.* In Modeling Approaches and Algorithms for Advanced Computer Applications, Studies in Computational Intelligence, pages 1–1. Springer, Cham, Switzerland, 2013. doi: 10.1007/978-3-319-00560-7_1.
[10] L.M. NI AND P.K. MCKINLEY. *A survey of wormhole routing techniques in direct networks.* Computer, 26(2):62–76, February 1993.
[11] N. KAVALDJIEV, G. J. M. SMIT, AND P. G. JANSEN. *A virtual channel router for on-chip networks.* In IEEE International SOC Conference, 2004. Proceedings., pages 289–293, Santa Clara, CA, USA, USA, September 2004. IEEE.
[12] G. DIMITRAKOPOULOS, A. PSARRAS, AND I. SEITANIDIS. *Microarchitecture of Network-on-Chip Routers.* Springer-Verlag New York, New York, NY, USA, 1 edition, 2015.
[13] H. SONG, B. KWON, AND H. YOON. *Throttle and preempt: a new flow control for real-time communications in wormhole networks.* In Proceedings of the 1997 International Conference on Parallel Processing (Cat. No.97TB100162), pages 198–202, Bloomington, IL, USA,, August 1997. IEEE.
[14] S. BALAKRISHNAN AND F. OZGUNER. *A priority-driven flow control mechanism for real-time traffic in multiprocessor networks.* IEEE Transactions on Parallel and Distributed Systems, 9(7):664–678, July 1998.
[15] S. L. HARY AND F. OZGUNER. *Feasibility test for real-time communication using wormhole routing.* IEE Proceedings - Computers and Digital Techniques, 144(5):273–278, September 1997.
[16] B. KIM, J. KIM, S. HONG, AND S. LEE. *A real-time communication method for wormhole switching networks.* In 1998 International Conference on Parallel Processing, 1998. Proceedings, pages 527–534, Minneapolis, MN, August 1998. IEEE.
[17] Z. SHI AND A. BURNS. *Real-Time Communication Analysis for On-Chip Networks with Wormhole Switching.* In Proceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip, NOCS '08, pages 161–170, Washington, DC,

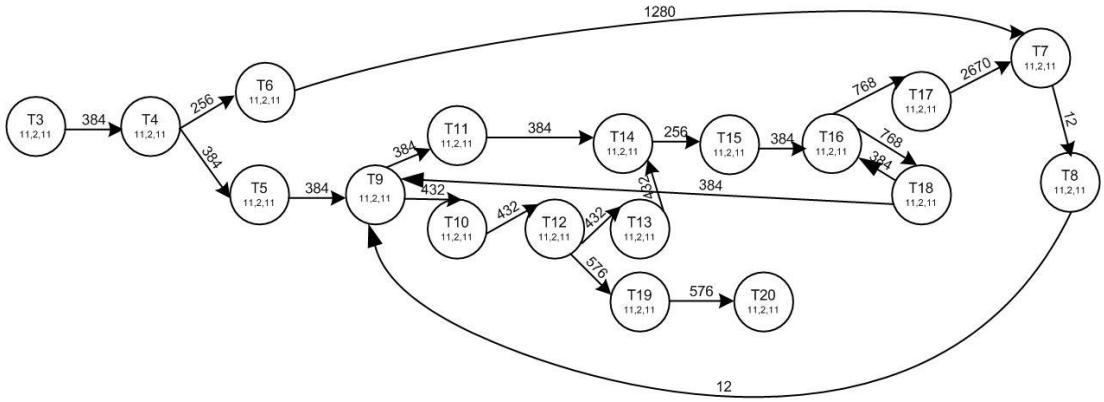


FIG. 5.11. *MPEG4 tasks graph.*

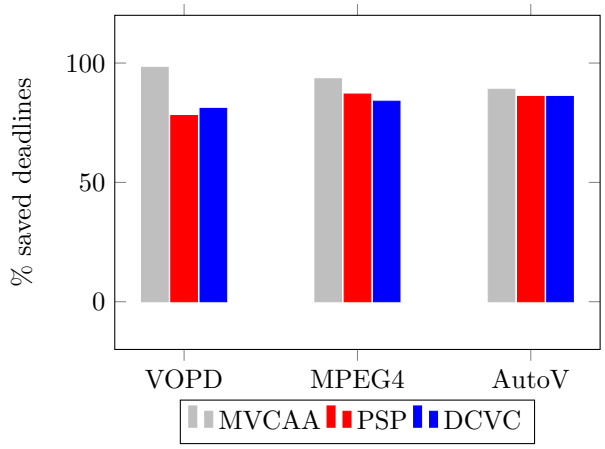


FIG. 5.12. *Percentage of saved deadlines on a 8×8 2D mesh NoC, 8 VC.*

USA, 2008. IEEE Computer Society.

[18] N. AUDSLEY, A. BURNS, M. RICHARDSON, K. TINDELL, AND A. J. WELLINGS. *Applying new scheduling theory to static priority pre-emptive scheduling.* Software Engineering Journal, 8(5):284–292, September 1993.

[19] A. MELLO, L. TEDESCO, N. CALAZANS, AND F. MORAES. *Virtual Channels in Networks on Chip: Implementation and Evaluation on Hermes NoC.* In 18th Symposium on Integrated Circuits and Systems Design, pages 178–183, Florianolpolis, Brazil, September 2005. IEEE.

[20] Z. SHI AND A. BURNS. *Real-Time Communication Analysis with a Priority Share Policy in On-Chip Networks.* In 2009 21st Euromicro Conference on Real-Time Systems, pages 3–12, Dublin, Ireland, Ireland, July 2009. IEEE.

[21] B. NIKOLIČ, H. I. ALI, S. M. PETTERS, AND L. M. PINHO. *Are Virtual Channels the Bottleneck of Priority-aware Wormhole-switched NoC-based Many-cores?* In Proceedings of the 21st International Conference on Real-Time Networks and Systems, RTNS '13, pages 13–22, Sophia-Antipolis, France, 2013. ACM.

[22] B. NIKOLIČ AND S. M. PETTERS. *EDF as an arbitration policy for wormhole-switched priority-preemptive NoCs #x2014; Myth or fact?* In 2014 International Conference on Embedded Software (EMSOFT), pages 1–10, Jaypee Greens, India, October 2014. IEEE.

[23] B. NIKOLIČ, S. TOBUSCHAT, L. SOARES INDRUSIAK, R. ERNST AND A. BURNS. *Real-time analysis of priority-preemptive NoCs with arbitrary buffer sizes and router delays* In Real-Time System Journal , 55(1):63-105,2019(Springer)

[24] B. SUDEV AND L.S. INDRUSIAK. *PFT : A low overhead predictability enhancement technique for non-preemptive NoCs.* In 2013 IFIP/IEEE 21st International Conference on Very Large Scale Integration (VLSI-SoC), pages 314–317, Porto Alegre, Brazil, October 2013. IEEE.

[25] B. SUDEV, , AND L.S. INDRUSIAK. *Low overhead predictability enhancement in non-preemptive network-on-chip routers using Priority Forwarded Packet Splitting.* In 2014 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), pages 1–8, Montpellier, May 2014. IEEE.

[26] J. DUATO, S. YALAMANCHILI, AND N. LIONEL. *Interconnection Networks: An Engineering Approach.* Morgan Kaufmann

Publishers Inc., San Francisco, CA, USA, 2002.

[27] Z. Shi, A. Burns, and L. S. Indrusiak. *Schedulability Analysis for Real Time On-Chip Communication with Wormhole Switching:.* International Journal of Embedded and Real-Time Communication Systems, 1(2):1–22, 2010.

[28] P. Kundu. *On-Die Interconnects for Next Generation.* In Workshop on On- and Off-Chip Interconnection Networks for Multicore Systems, California, 2006. Intel.

[29] T. Bjerregaard and S. Mahadevan. *A Survey of Research and Practices of Network-on-chip.* ACM Comput. Surv., 38(1):1–51, June 2006.

[30] M. Dehyadgari, M. Nickray, A. Afzali-kusha, and Z. Navabi. *Evaluation of pseudo adaptive xy routing using an object oriented model for noc.* In The 17th International Conference on Microelectronics., December 2005.

[31] M. A. Meghabber, A. Aroui, L. Loukil, A. El Hassan Benyamina, K. Benhaoua, and T. Djeradi. *A flexible network on-chip router for data-flow monitoring.* In 2017 5th IEEE International Conference on Electrical Engineering, pages 1–6. IEEE, October 2017.

[32] E. Bini. *Measuring the Performance of Schedulability Tests.* Real-Time Systems, 30(1-2):129–154, May 2005.

[33] S. J. Filho, A. Aguiar, F. G. de MagalhÃes, O. Longhi, and F. Hessel. *Task model suitable for dynamic load balancing of real-time applications in NoC-based MPSoCs.* In 2012 IEEE 30th International Conference on Computer Design (ICCD), pages 49–54, Montreal, QC, Canada, September 2012. IEEE.

[34] S. Murali and G. De Micheli. *Bandwidth-constrained mapping of cores onto NoC architectures.* In Automation and Test in Europe Conference and Exhibition Proceedings Design, volume 2, pages 896–901 Vol.2, Paris, France, February 2004. IEEE.

[35] D. Milojevic, L. Montperrus, and D. Verkest. *Power dissipation of the network-on-chip in a system-on-chip for MPEG-4 video encoding.* In 2007 IEEE Asian Solid-State Circuits Conference, pages 392–395, South Korea, November 2007. IEEE.

[36] S. Maatta, L. S. Indrusiak, L. Ost, L. Moller, J. Nurmi, M. Glesner, and F. Moraes. *Validation of executable application models mapped onto network-on-chip platforms.* In 2008 International Symposium on Industrial Embedded Systems, pages 118–125, France, June 2008. IEEE.

[37] Q. Xiong, F. Wu, Z. Lu, and C. Xie *Extending real-time analysis for wormhole nocs.* In IEEE Transactions on Computers, 66(09) pages 1532–1546, 2017. IEEE.

[38] Frederic Giroudot and Ahlem Mifdaoui *Buffer-Aware Worst-Case Timing Analysis of Wormhole NoCs Using Network Calculus* In 2018 IEEE Real-Time and Embedded Technology and Applications Symposium, Porto, Portugal, April 2018.

Edited by: Dana Petcu
Received: April 4, 2019
Accepted: June 5, 2019

DOI 10.12694/scpe.v20i3.1506
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 3, pp. 511–525. http://www.scpe.org

BLOCKCHAIN BASED E-CHEQUE CLEARANCE FRAMEWORK

NIKITA SINGH *AND MANU VARDHAN †

Abstract. This research work proposes a scalable and novel electronic cheque clearance framework. It is based on the blockchain where all banks willing to participate in this system must join the proposed blockchain based framework in order to provide the faster cheque clearance facility to its customers. The proposed e-cheque system is free from the various security attacks such as alteration of the e-cheque, double spending of e-cheque, counterfeits e-cheques. The e-cheque generated in the proposed system can be deposited electronically or physically via teller machines. The proposed system is highly scalable because on an average only 32.2% of nodes participate in the proposed trust based consensus mechanism and further message exchange per consensus process is much lesser as compared to PoW approach.

Key words: e-cheque, blockchain, scalable trust based consensus mechanism, multithreaded parallel transaction search,

AMS subject classifications. 68M14, 68M12

1. Introduction.

Advancement in technology has brought remarkable changes in all the sectors such as financial, industrial, education, administration etc. Banking sector has kept pace in adopting these technological shifts and has grown by leaps n bound. The major transformation of the banking sector took place in late 80's or early 90's. During this decade, card based payment system and electronic clearing system (ECS)[2] to transfer money from one bank account to another electronically were introduced. In later decades, Real Time Gross Clearance (RTGS), NEFT (National Electronic Funds Transfer) were also included in banking system. The financial institutions have introduced cheque truncation system (CTS) due to large volume of transactions for faster cheque clearance. In CTS, a Magnetic Ink Character Recognition (MICR) [11] coding is printed on all the cheques which are read by the MICR readers and the system automatically detects the drawer's bank and branch by scanning this MICR code. Cheques are transferred electronically (scanned images of cheques) to the drawer's bank. This process reduces the cheque clearance time. Gjomemo et al. [8] discusses various ways of forgery in digital cheques such as replacing the duplicate signature of any person, changing the precision in cheque amount by using digital image processing techniques. Rajendra and Pal [16] propose digital watermarking based approach for detection of any forgery in cheque. Anderson [1] proposes architecture of the e-cheque framework. Chang et al. [5] propose an e-cheque system that is based on mutual authentication of drawer and payee. Blockchain based large e-governance application such as blockchain based property transaction system [20] are gaining attention of researchers.

1.1. DLT, Bitcoin and Blockchain.

Digital Ledger Technology (DLT) and blockchain have been used interchangeably since the concept of the bitcoin cryptocurrency system was introduced by Satoshi Nakamoto in 2008 [14]. DLT encompasses the blockchain and other type of distributed ledgers and the records are distributed as a chain of blocks across a peer-to-peer network. All the transaction in the blockchain is recorded in the form of chain of blocks. Each block contains a unique header which is cryptographically computed and this feature attributes the immutable nature of the blockchain and this hash commits to the header of the previous block. Before a transaction is committed to the ledger, it has to be agreed upon by the active participants of the network in order to guarantee the trustworthiness of the information being incorporated into the blocks. This is where the distributed ledger consensus protocols become important and determines which state of the database is chosen to be valid and true. It is only when consensus is achieved that the new transaction is recorded into the block and is linked to the already existing chain of blocks using a hash pointer to the previous block.

1.2. Novelty of the proposed Approach.

This paper proposes a Digital Ledger Technology (DLT) based solution to the cheque clearing system for banking transactions. Presently, some banks provide e-cheque facilities to their customer but the scope of e-cheque is limited to its own banking branches only since e-cheque issued by a bank cannot be deposited in other bank due to security and authentication issues. The proposed approach extends the scope of e-cheque from local to global banking and analyzes the vulnerabilities of e-cheque

*CSED, National Institute of Technology Raipur, India (nikitasinghk@gmail.com)
†CSED, National Institute of Technology Raipur, India (mvardhan.cs@nitrr.ac.in)

against double spending and forgery. The analysis reveals that proposed e-cheque system is not vulnerable to these threats. The proposed system is based on permissioned blockchain and is intentionally designed in such a way that any bank can see the cheque issued by its customer or deposited by any other bank. This enables a bank to validate the deposited e-cheque. Further, the information about the any customer such as personal details, balance information and frequency of transaction remain confidential. The proposed system only stores the issued and deposited cheque information into the blockchain. Aggregated balance of any customer is not visible to any other bank or its miners. Further a scalable trust based consensus mechanism ensures that increase in number of transactions shall not degrade the performance of the proposed system.

1.3. Organization of the Paper. A brief literature survey of leading research papers that concerned this proposed approach have been researched in section 2. Section 3 has 6 subsections that detail the proposed approach. Section 3.1 proposes network architecture for blockchain based e-cheque system. Section 3.2 proposes blockchain based e-cheque generation process. Section 3.3 proposes blockchain based e-cheque payout process followed by 3.4 that proposes consensus mechanism & leader election process along with analysis of results. Section 3.5 proposes multithreaded parallel transaction search algorithm followed by 3.6 that analyzes security threats & mitigation in the proposed approach.

2. Literature Survey of Leading Related Work . The global financial crisis that occurred in 2008 imposed strict and rigid banking norms and regulations worldwide with the view to prevent and deflect a crisis like this to ever happen again. Nguyen [15] attempts to bring into focus the role of blockchain technology in the development of a much more customer- centric and transparent banking system. Barclays becomes the first industry to adopt blockchain technology for its business [4]. Santander [17] also started to use blockchain technology for real-time trade transactions. InsurChain [10] is first blockchain application for insurance ecosystem. Starbase [21] also started to use crypto-tokens for crowd funding from various sources. Guo and Lang [9] in their paper describe how blockchain technology is the combination of several other existing computer technologies namely, distributed data storage, peer to peer systems, distributed consensus mechanism, and encryption algorithms. Cocco et al [6] in their paper talk about the sustainable development and potential of blockchain as a banking technology by taking the bitcoin system under consideration. Eyal [7] discusses the role and potential of the blockchain technologies to fulfill the requirements. The authors in [13] study the various applications of the core bitcoin protocol and conduct an experiment to ensure whether the blockchain can be operated in a secure environments and networks.

The consensus process is responsible for selection of the leader for mining the new block, verifying the transaction in new block and achieving the consensus of other miners on new block before adding the block into blockchain. There exists various consensus mechanism to handle the byzantine failures such as Proof-of-work (PoW) [14], Proof-of-Stake (PoS) [12] etc. The PoW [14] handles the issues of Byzantine Generals Problem by imposing a puzzle to miners. The miners have to solve the puzzle in order to get the opportunity for elected as leader and mine the block. The new block is added to blockchain when majority i.e. 51% votes of miners are garnered. In PoS, the highest stake holder miner always get chance to mine the new block and other miners achieve the consensus on the new block

In any peer-peer systems or distributed systems, trust of nodes also plays an important role in selection of most efficient and secure node selection for various operations. Bano et al. [3] state that the important factor that distinguishes blockchains from traditional distributed databases is the ability to operate in a decentralized setting without relying on a trusted third party. Schwartz et al. [18] are of the opinion that several consensus algorithms exist for the Byzantine Generals Problem, few of which are suitably designed for decentralized and distributed payment systems. Tschorsch & Scheuermann [22] state that pioneering contributions of the virtual currency Bitcoin is achieving the degree of decentralization which was previously thought unachievable. Singh et al. [19] are of the view that each bank has to maintain a huge data center with expensive skilled manpower requirements and these data centers consume large energy, thus contributing to increased carbon emission.

3. Proposed DLT Based E-Cheque System. This research work proposes a novel and comprehensive electronic cheque transactions framework. The e-cheques generated by the system can be deposited to the bank either electronically or physically. The proposed system is based on the blockchain technology; hence all banks willing to implement the proposed system must join the proposed blockchain based framework in order

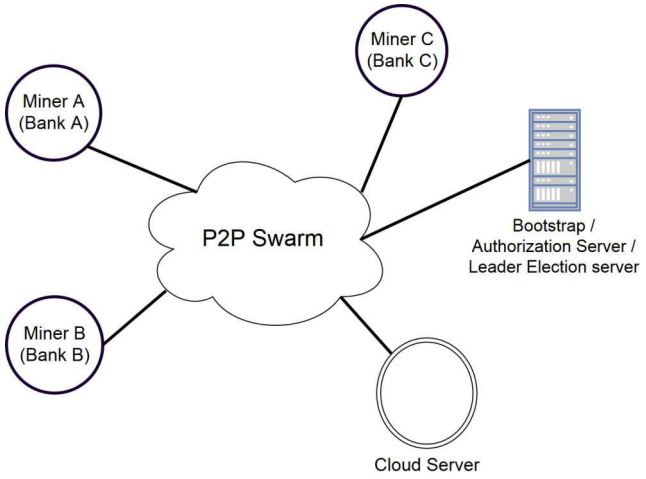


FIG. 3.1. *Network architecture of DLT based e-cheque framework*

to provide the e-cheque facility to the customers.

3.1. Proposed Network Architecture for e-Cheque System. The network architecture of the proposed system comprises of entities such as different participating banks and their respective web servers that are replicated, miner nodes for each of these replicated web servers, cloud data center and some professional miners that may also be engaged as these miners carry state of the art hardware resources. All the miners are connected together with a common p2p swarm network as shown in figure 3.1 and a common blockchain exists that is used by all these participating banks. Each bank provides an interface for e-cheque generation and e-cheque deposit through online portal. The teller machine fetches information from miner of the bank and cloud data centre. Teller machines shall scan the barcode and read the e-cheque that is being deposited by any customer. All the e-cheques generated and deposited by the customers will be stored in the closed blockchain in the form of transactions.

A bootstrap server maintains the list of authorized miners. To join the p2p swarm network, each miner connects with bootstrap server by sending a request to join the p2p network. On receipt of any request from miner, bootstrap server replies back with the list of active miners as the response of the request. Now the incoming miner is able to connect with the all other active miners. Hence, p2p swarm is formed through the bootstrap server. This bootstrap server is also part of the p2p swarm network and participates in leader election process as discussed in section 3.4. Each bank may have multiple miners as shown in figure 3.2 where mulitple miners of a bank are connected to the all servers of the bank. The miner local to a bank is called internal miner and these miners are connected to bank server via internal private network of that bank. The internal miners are connected to other bank miners via p2p swarm network. Master and secondary server handle banking application along with the role of web server. The next section discusses the process of the e-cheque generation, when any customer has to issue a cheque in favor of some other entity. It is important to note that two major activities of the proposed system for storing e-cheque transactions in blockchain are verification of:

i. e-cheque issued &
ii. e-cheque clearance.

3.2. Proposed DLT based e-Cheque Issue. In the proposed system, for e-cheque issue, the drawer generates e-cheque from online banking portal of the bank. In the proposed system, each customer is issued with a pair of public and private keys and to generate the e-cheque, customer needs to digitally sign each transaction using his private key. The public key of all customers of all the banks is known to all miners. The generated e-cheque has unique barcode and its number printed on it. During verification of this newly created e-cheque, the digital signature of the drawers is verified by at least two internal miners. Hence, the server multicasts this transaction to two least loaded miners to verify this transaction. Upon verification of digital signature,

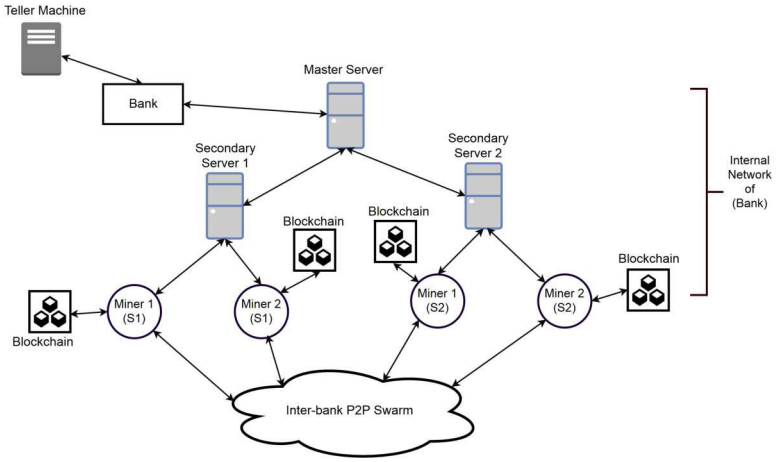


FIG. 3.2. *Intra-Bank p2p network and Banking server architecture*

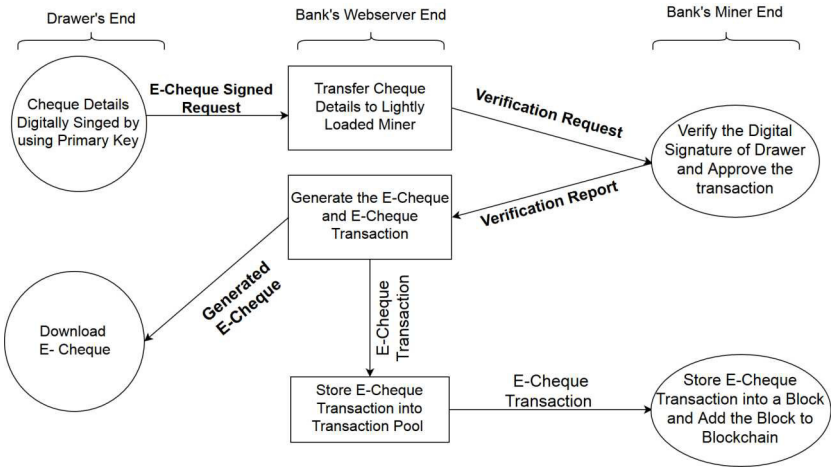


FIG. 3.3. *e-Cheque Issue Process*

the transaction is added to transaction pool and verified e-cheque is generated as discussed in section 3.2.1. The banking portal now allows the drawer to download this e-cheque as valid e-cheque. Set of these verified transactions are stored in a block by an internal miner. Figure 3.3 illustrates the process at each end of the proposed system along with work flow of e-cheque generation request when any account holder requires a cheque for making any payment. This verified e-cheque is downloaded by drawer and may be sent electronically (mail / sms etc.) to the payee. Payee can download this e-cheque sent by drawer and deposit the same physically into its own bankers teller machine or electronically by payees banking portal.

3.2.1. Transaction format for e-cheque Issue. Before all the verified e-cheques issued by various entities can be cleared, these verified e-cheques should be recorded into the blockchain in form of transaction so as to validate whether these e-cheques are eligible for being honored. Eleven different attributes that constitute an e-cheque are:

$Request\_Type$: type of the transaction, {set to value to "e-cheque issue"}
B: the unique barcode number assigned to the e-cheque,
D_N: name of the drawer,
B_N: name of the drawer's bank,
B_B: name of the bank branch where the drawer's account exist,

D_A: drawer's account number,
C_{qn} : cheque number,
C_{qt}: is the cheque type i.e. banker's cheque, account payee cheque etc.,
P_N: name of payee,
A_t: amount and
C_{qd}: cheque issue date

These e-cheque attributes are defined by a set EC:

$$EC = \{Request\_Type, B, D_N, B_N, B_B, D_A, C_{qn}, C_{qt}, P_N, A_t, C_{qd}\}$$

To secure the set EC, secure hash of this set is also computed before the set EC is digitally signed by the customer. $SHA256$ algorithm is used to compute the hash of this set EC:

$$Hash_{EC} = SHA256(EC)$$

Drawer signs the set EC and hash of this set EC with its private key. Drawer's private and public key are represented by DS_K and DP_K. The digital signature is obtained using $ECDSA$ algorithm as:

$$digital\_signature = ECDSA(DS_K, Hash_{EC}, EC)$$

After obtaining the digital signature, the same is verified by internal miners and a copy of verified e-cheque is generated and provided to the customer. At the server side, hash of the generated e-cheque is also recorded into the transaction. The generated e-cheque is represented by a file 'E' and the hash of this file is computed as:

$$Hash_E = SHA256(E)$$

Now the complete transaction is represent by set T_X as:

$$T_X = \{EC, Hash_{EC}, digital_signature, Hash_E\}$$

This transaction is stored in the global transaction pool and later placed into the block during the mining process.

3.2.2. Block Creation for e-Cheque Issue Transactions. In the proposed framework, a block contains only one kind of transaction based on Request Type. This is because during e-cheque generation, only internal miners shall perform verification whereas for payout, all the miners participate in verification of details of e-cheque payout. Hence, the proposed blockchain has two types of blocks i.e. the block that contains transactions with "Request_Type" as "e-cheque issue" and the blocks that contain transactions having "Request_Type" as "e-cheque payout". This is defined in block_type field in each block.
All the attributes listed in Sect. 3.2.1 require about 1400 bytes of storage. Hence this implementation has been done with a block size of 50 KB with each block containing 30 transactions. Selection of the miner for mining the new block is always controlled by the consensus algorithm. Before this block becomes part of blockchain, miners of all the participating banks only verify the digital signature of this miner during consensus process.
The block generated by the miner contains following attributes:

- hash of previous block,
- timestamp,
- hash of all the transaction,
- list of the transactions and
- digital signature of the miner.

The hash of previous block is defined by PB_{Hash}, and set of the transaction is defined as T where:

$$T = \{T_{X1}, T_{X2}, T_{X3}, ..., T_{Xn}\}$$

Each T_X represent the transaction defined in previous section. Hash of the transaction is computed as:

$$Hash_T = SHA256(T)$$

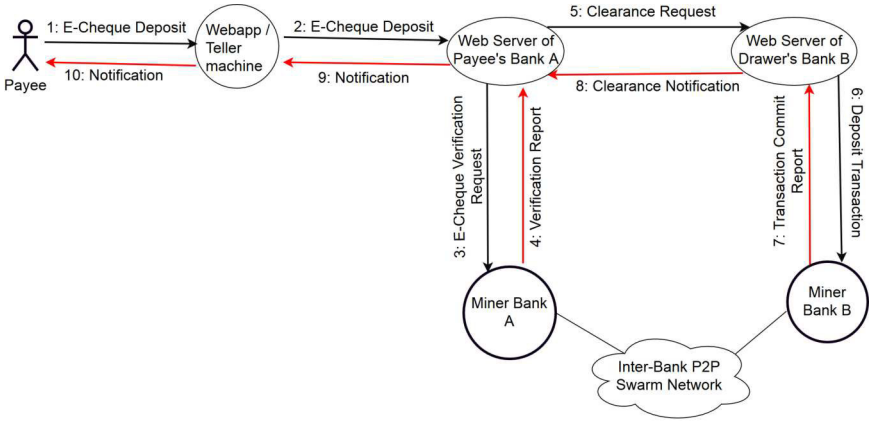


FIG. 3.4. *E-Cheque deposit and clearance process in the proposed system*

The value of attribute block_type is set to "e-cheque issue" as all the transactions stored are for issuing an e-cheque. The time instance when a block is created is denoted by TS. Finally the miner has to sign the all the transaction with its private key. Let the private and public key of the miner be represented by MS_K and MP_K. Finally, digital signature is obtained using $ECDSA$ algorithm as:

$$M_{digital\_signature} = ECDSA(MS_K, Hash_T, block\_typeT, TS)$$

The content of the block is represented as set B_L where:

$$B_L = \{PB_{Hash}, TS, Hash_T, block\_type, T, M_{digital\_signature}\}$$

This newly created block is broadcast to all the miners of every bank to achieve consensus only on digital signature of miner which create the block using respective public key. This is necessary to ensure that the block is being generated by authorized miner and block is added to the block in their blockchain if found valid.

3.3. Proposed DLT based e-Cheque Clearance. The payee can deposit the e-cheque electronically through the online banking portal or physically by depositing the print copy of the e-cheque in the teller machine. The server accepts this e-cheque and performs a search operation for corresponding transaction on blockchain for verifying validity and authenticity of it. The e-cheque clearing request is approved by the banking system once the validity and authenticity of the e-cheque is proved; otherwise it is rejected. In physical deposit process, teller machine scans the barcode of the e-cheque and extracts the respective transaction corresponding to this cheque that is stored in the blockchain with block type "e-cheque issue". The clearance request is generated by the teller machine after verification of the e-cheque. Once the e-cheque is cleared by the system, the details of the e-cheques are again stored in blockchain in the form of the transaction in block type e-cheque payout. Figure 3.4 shows the process flow of e-cheque deposit request and clearance process. Once the payee's bank server receives e-cheque deposit request during clearing process, the request is forwarded to the miners in the p2p network. These miners search for corresponding transaction in the blockchain and verify its validity and authenticity. The search is based on the proposed Multi-threaded Parallel Transaction Search Algorithm (MPTSA) that reduces the search time considerably. The payee's bank server generates the clearance request to the drawer's bank on valid e-cheques otherwise it rejects the request and notifies the customer accordingly.

3.3.1. Transaction format for e-cheque clearance. To ensure that only valid e-cheque get deposited and used only once, the proposed system generates a transaction for each e-cheque issued. The miners first search the corresponding e-cheque transaction in the blockchain and generate the validity report for the e-cheque based on its attributes. The transaction that matches the requested attributes and has newest timestamp value is picked up for validity check. The requested e-cheque would be valid only when the value of Request_Type" field of searched transaction is "e-cheque issue" else e-cheque is considered as invaild.

The value of attributes "Request_Type" is set to the "e-cheque payout" as this transaction is prepared for commit operation. The attributes defined for the e-cheque deposit request are:

$Request\_Type$: type of the transaction, {set to value to "e-cheque payout"}
S_{id}: Clearance request identifier,
B, D_N, B_N, B_B, D_A, C_{qn}, C_{qt}, P_N, A_t, C_{qdt} are the attributes that are same as defined in section 3.2.1.
C_{qdd} : cheque deposit date,
P_{BN}: name of the payee's bank,
P_{BB}: name of the payee's bank branch, name

All the above attributes are part of the set P. Therefore, the e-cheque deposit attributes are represented by set EC.

$$EC = \{Request\_Type, S_{id}, B, D_N, B_N, B_B, D_A, C_{qn}, C_{qt}, P_N, A_t, C_{qd}, C_{qdd}, P_{BN}, BB\}$$

To secure EC, secure hash of this set is computed using $SHA256$ before the set EC is digitally signed by the payee's bank server which initiates clearance request.

$$Hash_{EC} = SHA256(EC)$$

Now, payee's bank server signs the set EC and hash of the set EC with its private key. The payee's bank server's private and public key is represented by SS_K and SP_K respectively. The digital signature is obtained using $ECDSA$ algorithm as:

$$digital\_signature = ECDSA(SS_K, Hash_{EC}, EC)$$

Finally, the complete transaction is represented by set T_X as:

$$T_X = \{EC, Hash_{EC}, digital\_signature\}$$

The clearance request is only approved by the drawer's bank server when the generated e-cheque payout transaction request is stored in the blockchain. All the valid e-cheque clearance transactions are added in the block before these become part of block chain. This is elaborated in the next section.

3.3.2. Block Creation for e-Cheque Clearance Transactions. To store these transactions into blockchain, leader miner creates a block and adds verified transactions that has "Request_Type" value as "e-cheque payout" and sets the "block_type" field to "e-cheque payout". To add this block in the blockchain, all peers must verify all transactions in the newly created block. Once all the transactions of newly created blocks are verified by each miner, the consensus process is started to obtain the final consensus to add this block into blockchain. A novel approach for consensus mechanism is proposed in the next section.

3.4. Proposed Scalable Trust Based Consensus Approach. The proposed e-cheque transactions framework comprises of two types of miners; one that are part of the bank and the other being outsourced or private. The nodes that are part of the banking system are termed here as Banking System Miners (BM). The other are Authorized Professional Miners (AM) that have investments in state of the art infrastructure (farms) and offer their services so as to encash their investments in these farms. Based on the number of transactions, we classify BSM into Heavily loaded BSM (ROBM) and Lightly loaded BSM (RLBM).

3.4.1. Leader Election Process. To maintain the blockchain consistency, the process of block mining needs to be synchronized. The leader election mechanism holds this responsibility and by synchronizing mining process, consistency in blockchain is maintained. The leader election mechanism elects a leader miner among several miners for mining process for each block. This leader miner mines the new block and broadcasts it to all miners for consensus process. In the proposed system, the bootstrap server maintains the list of all active miners. Hence, allocation of mining slots to miners is handled by bootstrap server.

TABLE 3.1
Proposed table structure to maintain the miner's status

Node ID	Computational Resources	CPU Load	CPU Load Status	Trust Value
4 Byte	Memory, CPU	2 Bytes	1 Byte	1 Bytes

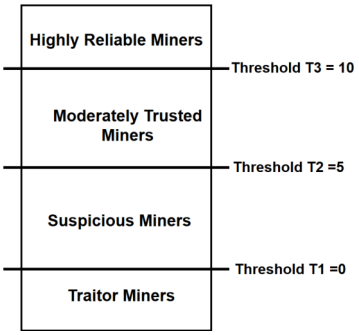


FIG. 3.5. *Thresholds for classification of trust value of miners*

3.4.2. Proposed Trust based Consensus Algorithm (TCA). Most of the existing blockchain applications demand 51% of votes in order to add a block to an existing blockchain. This can be very demanding when the application being developed involves real time transaction processing. To overcome these issues, a hybrid efficient consensus mechanism based on the load of the node and its trust value is proposed here. Objective of proposed consensus algorithm is to reduce the overhead of message exchange and time required to achieve the consensus. In the proposed approach, each miner maintains the status table of other miners as shown in table 3.1.

A. Assigning Trust to Miners / Nodes. In order to select few reliable miners for participating in consensus mechanism, we compute the Trust Value (TV) of all these nodes based on the following three attributes:

i. Computation Resources (CR) available at each node,
ii. Response Time (RT) of each node along with its communication (bandwidth) time &
iii. Trustworthiness based on historical Correctness of Transaction (CoT) verification done earlier during any new block addition process.

Computation of Trust Value (TV) is defined by following conditions:

i. If CR is state of the art at any node & RT is ¡ 30 ms, its CR is set to 1, else set to 0.
ii. If a node is connected by a high bandwidth link, its RT is set to 1 else 0.
iii. If RT lies between 30 & 60 ms, it is set to 0.5. process.
iv. When correct verification done by a miner, CoT is incremented by 1 else decremented by 5. This is because there is no scope for malicious / incorrect transaction.

Trust value of any node is computed as:

$$TV = CR + RT + CoT$$

This trust value is broadcast to all the nodes / miners in the system. This initial setup is done when any node / miner joins the p2p swarm. Each node maintains a list indexed on following attributes:

i. Load of BSM sorted on the load. Nodes above a certain threshold are designated as HBSM else LBSM.
ii. Further the same list is sorted based on the value of TV
iii. List of private miners is sorted on the TV score. TV of PMs above a certain threshold are designated as highly reliable else trusted / suspicious miners.

In the proposed system, the miners are categorized into four different groups based on their trust Value (TV). The categorized five groups are:

- G1: Highly reliable ROBM,

TABLE 3.2
Agent vote during Consensus Process

Hash of New Block	Favorable Agents			Not Favorable Agents		
	CN ID	**Response Time**	**Group**	**CN ID**	**Response Time**	**Group**
Hash Value	ROBM1, ROBM2, ROBM3,	T1, T2, T3,	G1	ROBM5, ROBM8, ROBM9,	T1, T2, T3,	G1
	RLBM1, RLBM2, ...	T4, T5, ...	G2	RLBM7, RLBM 6,	T4, T5,	G2
	RAM1, RAM2, RAM3, ...	T7, T8, T9	G3	RAM4, RAM4, RAM8, ...	T7, T8, T9	G3
	MRAM1, MRAM2, ...	T10, T11, ...	G4	MRAM7, MRAM9, ...	T10, T11, ...	G4

- G2: Highly reliable RLBM,
- G3: Highly reliable Private (Authorized) Miners RAM,
- G4: Moderately reliable RLBMs & RAMs,
- G5: Un-trusted miners or other miners

G1, G2 and G3 are the groups of miners that achieve trust value above 10 whereas in group G4, the miners with trust value between 5 to 10 are included. The miners that have trust value below 5 are classified under the G5 group as shown in figure 3.5. These miners will never get chance for being selected as consensus agents as discussed in subsection B. So this proposed method reduces the overhead of broadcasting a new block reduces by more than 50%. This again saves computation time and network bandwidth.

B. Role of leader in Consensus Mechanism. The consensus mechanism discussed above uses multicast instead of the broadcast thus ensuring the scalability of the proposed system. The selection of the miner's for verification of the newly created block is responsibility of the leader miner based on the TV. Each miner in the network maintains a miner node status table. The leader selects randomly 25% miners from G1 group, 25% miners from G2 group, and 50% miners from G3 and 25% of G4 groups are selected. These selected nodes are called consensus agents. These consensus agents verify the new block and broadcast their votes to all the peers on newly created block.

C. Role of consensus agents. The consensus agents receive the newly created block from the leader. The consensus agent verifies all the transactions stored into the block and the digital signature of the leader. After the verification process, consensus agent broadcast its consensus on newly created block to the all peers in the network.

D. Role of the other miners. The miners including consensus agents receive the newly created block from the leader and store it to the temporary buffer. Now all the miners wait for the consensus votes of the consensus agents. Each miner maintains miner node status table; hence, each miner waiting for consensus, can identify the consensus agents. The miners also modify the trust value of the consensus agents based on the votes and trust management policy as discussed in subsection A. This table records the list of those agents that are in favor of adding the block meaning thereby that the block is valid (all transactions listed in block are verified and authentic) and list of the agents those who are not in favor of the block meaning that the block is invalid as shown in table 3.2. All the Miner stores the votes of the agents into this table. On receipt of votes from all the agents, each miner computes the final consensus on newly created block based by the counting of votes. It is proposed that minimum 10% of the votes among nodes of G1, 41% of G2, 51% of G3 and 25% nodes from G4 are required in order to achieve final consensus as shown in figure 3.6.

This ensures the following:

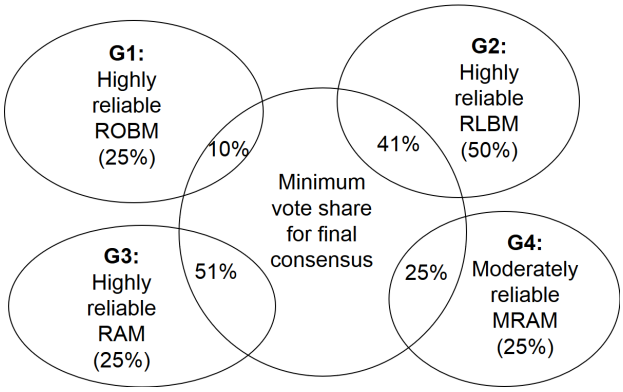


FIG. 3.6. *Vote share for final consensus*

i. Majority vote among BM is achieved although only few reliable nodes participate,
ii. This multicasting also reduces network load &
iii. Even if all the RAMs collude, these nodes cant hijack the consensus mechanism.

Each miner including leader and consensus agents vote for the final consensus. This final consensus decides whether the block should be added to blockchain or not. The leader notifies the block status to the bank server that generates the e-cheque clearance transaction.

E. Analysis of Proposed Trust based Consensus Mechanism. Analysis of the proposed TCA is carried out in order to establish its validity and robustness. Further, the results obtained are compared with the widely used PoW [20] algorithm based on the following parameters:

i. Number of Messages Exchanged,
ii. Time required to achieve consensus &
iii. Analysis of CPU, Memory & Network Utilization of Consensus Node (CN).

During the experiment, all these parameters are recorded and analyzed when a new block is being created and broadcast to all miners for verification of transactions stored in it.

Number of Messages Exchanged. The performance of the proposed TCA consensus approach is measured in terms of number of messages required as shown in table 3.3 to achieve the consensus. In traditional approach, all 'N' miners participate in consensus process and broadcast their consensus to all 'N-1' nodes. This causes the overhead in network as total N(N-1) messages are exchanged. In the proposed approach, fewer numbers of nodes are selected on the basis of respective trust value only and these selected nodes participate in the consensus mechanism. During different simulations, results are recorded with increasing number of miner nodes (N) and its impact on the total number of messages exchanged in order to achieve consensus.

Let, the total number of miners are N. Among these N, let, total number of G1 miners be $g1$, total number of G2 miners be $g2$, total number of G3 miners be $g3$ and Total number of G4 miners be $g4$. Hence $g1+g2+g3+g4 = N$. Let the total consensus agents selected from G1 group be defined by $a1$ as $(25*g1)/100$, $a2$ as $(50*g2)/100$, $a3$ as $(25*g3)/100$ and $a4$ as $(25*g4)/100$. Hence Total number of Message Exchange (TME) require in consensus process are:

$$TME = (a1 + a2 + a3 + a4)(N - 1)$$

The minimum number of consensus message (MFC) required in achieving Final_Consensus is:

$$MFC = \{(a1/10) + 1 + (2 * a2/5) + 1 + (a3/2) + 1\} * (N - 1)$$

Total message exchange required in proposed approach are also compared with the traditional approaches as shown in Table 3.3. The analysis of results listed in table 3.3 and 3.4 revels that the proposed approach requires on an average only 32.2% of message exchange per consensus process as compared to traditional PoW approach. The proposed trust based consensus mechanism requires minimum 23.66% MFC from trusted miners to achieve the consensus.

TABLE 3.3
TME and MFC analysis in varying number of nodes in network

S.No.	N	g1	a1	g2	a2	g3	a3	g4	a4	TME	MFC
1	100	30	8	30	15	20	5	20	5	3267	1386
2	200	50	13	50	25	50	13	50	13	12736	4378
3	300	70	18	70	35	100	25	60	15	27807	9867
4	400	100	25	120	60	100	25	80	20	51870	18753
5	500	120	30	150	75	130	33	100	25	81337	28942
6	600	140	35	180	90	160	40	120	30	116805	36539

TABLE 3.4
TME and MFC comparison of traditional and proposed approach

S.No.	N	PoW		Proposed Approach		Message Reduction Proposed Approach (%)	
		TME	MFC	TME	MFC	TME	MFC
1	100	9900	4951	3267	1386	33	27.9
2	200	39800	19901	12736	4378	32	21.9
3	300	89700	44851	27807	9867	31	21.9
4	400	15960	79801	51870	18753	32.5	23.4
5	500	249500	124751	81337	28942	32	23.2
6	600	359400	179701	116805	36539	32.5	20.3

Time required to achieve Consensus. In this experiment, time to achieve the consensus on same block is recorded for the proposed approach and proof-of-work approach. In each iteration, the number of miners is increased by 100 with consensus nodes (CN) being constant. From table 3.5, it can be observed that the proposed trust based consensus mechanism requires fewer consensus nodes nodes as compared to PoW for achieving consensus. Coupled with this benefit is atleast 25% lesser time requirement for adding any new block.

3.5. Proposed Multithreaded Parallel Transaction Search Algorithm (MPTSA). In any blockchain application, among other factors, the time for consensus on any new block also depends on the number of transaction placed in new block. This is because each miner has to traverse the blockchain in order to verify these new transactions. Hence, the deeper the blockchain traversal required, higher the time required to verify the transactions. To reduce the verification time, this paper proposes a multithreaded parallel transaction search algorithm. This algorithm traverses the blocks in parallel by using kernel level threads. Searching a transaction in blockchain involves traversing blockchain sequentially and comparing each transaction details with the attribute of transaction being verified. To reach any predecessor block, the hash value of that block that is stored in its successor block is used. The retrieved block contains list of transactions and hash value of its previous block. In the proposed approach, certain number of kernel level threads is used to achieve the parallelism in tasks such as retrieving a block and comparing the transactions. One of the threads gets placed at previous block while all other threads perform read and comparison operation as shown in figure 3.7. This causes parallel processing of transaction comparison task along with advance block retrieval. For example, if a block contains 5 transactions in any blockchain, then the proposed approach searches for the transaction with 6 threads such that one thread always works for retrieving the contents of previous block and remaining five threads perform transaction comparison operation for five transaction per block. This will enhance the overall performance of the searching time with multi-core processing capability. Figure 3.7 shows the illustration of parallel search execution of task.

3.6. Analysis of Proposed Multithreaded Parallel Transaction Search Algorithm (MPTSA). To verify the performance of proposed MPTS algorithm, experimental setup carried out in java on machine having CPU configuration as Intel i7-4790 @ 3.60 GHz, RAM 8GB DDR3 (1600 MHz), Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless, Storage: 100GB. In this experiment, random query is fired and search

TABLE 3.5
Comparison of consensus achieving time of proposed approach with PoW

S.No.	No. of. Miners	PoW		Proposed Approach	
		CN	Time (Sec.)	CN	Time (Sec.)
1	100	100	4.067	33	2.962
2	200	200	8.265	64	6.033
3	300	300	9.512	93	6.941
4	400	400	13.057	130	9.532
5	500	500	16.672	160	12.170
6	600	600	20.302	195	14.117

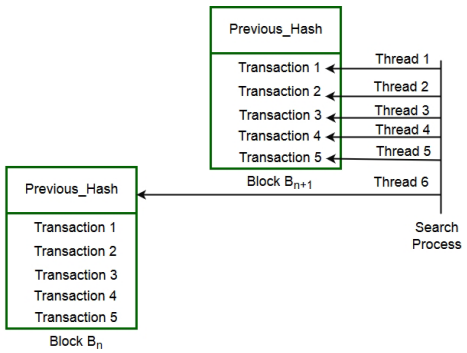


FIG. 3.7. *Proposed parallel transaction search*

time of the proposed approach with different number of parallel thread is obtained. The overall experiment is performed in two scenarios. In first scenario, length of the blockchain is kept 1500 blocks and size of block is 40 KB. In second scenario, the length of blockchain is kept 2000 blocks and size of each block is fixed to 400KB. In both scenarios, the search time of the approach is recorded for 1, 4, 8 and 16 threads as shown in tables 3.6 and 3.7.

For six different simulations, the average time reduction for searching any cheque transaction details is more than 60% on an average. Hence the proposed MPTSA shall lead to faster clearance of the pending cheque transactions.

3.6.1. Scalability of the Proposed Approach. Total time required to obtain consensus on one block containing 30 transactions as listed in table 3.5 is 2.96 sec for completion. The average transaction search time from a blockchain consisting of 2000 blocks is 1.56 seconds as listed in table 3.7. In the proposed e-cheque framework where a block contains 30 transactions, validating these transactions for e-cheque payout requires (2.96+ (1.56*30)) = 49.76 seconds with an octa-core machine. So on an average, the proposed framework requires 1.65 seconds to clear an e-cheque. Hence, the proposed e-cheque transaction framework is suitable to be deployed in real time banking and increase in number of transaction shall not degrade the performance of this system, making it scalable.

3.7. Vulnerability Analysis of the Proposed Approach . The digital documents are always vulnerable to alteration, threat of being counterfeit etc. Apart from this, customer may create multiple copies of digital document; hence vulnerability of the issued e-cheque for alteration and double spending problem needs to be analyzed. This section discusses the inherent capability of proposed system to handle such security threats.

3.7.1. Handling the Threat of Alteration of E-cheque. In the proposed system, the e-cheque issued by any drawer is always recorded in the blockchain in the form of a transaction. The blockchain is stored on nodes that are residing in different sites and connected through decentralized p2p network. Proposed system is able to detect altered e-cheques at the time of deposit of e-cheque because each deposit operation requires

TABLE 3.6
Search time of MPTSA on Blockchain of 1500 blocks

S.No.	No. of Block Traversed	Block search time (in sec.) using different number of threads			
		1	4	8	16
1	1465	3.66	2.78	2.17	1.33
2	1200	2.89	2.17	2.01	1.19
3	1498	3.91	2.99	2.34	2.11
4	1342	2.43	2.08	1.98	1.31
5	1481	3.78	2.86	2.08	1.34
6	1235	3.08	2.18	2.00	1.21

TABLE 3.7
Search time of MPSA on Blockchain of 2000 blocks

S.No.	No. of Block Traversed	Block search time (in sec.) using different number of threads			
		1	4	8	16
1	1678	3.38	1.83	1.76	1.06
2	1702	3.92	2.11	1.79	1.15
3	1812	4.29	2.42	1.89	1.28
4	1894	4.57	3.01	2.17	1.97
5	1949	5.31	3.98	3.01	2.24
6	2000	7.61	3.45	2.95	2.34

consensus of miners as discussed in section 3.4. Figure 3.8 explains the detection and rejection process of any altered e-cheque by the proposed system.

3.7.2. Handling Threat of Double Spending of e-cheque. The e-cheque issued to any payee may be deposited in multiple banks by the payee. As discussed in section 3.3 that elaborates Proposed Blockchain based e-cheque Payout Process, when a payout is achieved at one bank, then during subsequent payout process, the miner during the consensus process shall detect the attribute request_type as being set to e-cheque payout in "block_type" field to "e-cheque payout" as illustrated in figure 3.9. Hence, the proposed system shall not achieve consensus for this second payout of e-cheque. Hence the proposed framework prevents double spending problem.

Once a request of e-cheque is committed in blockchain, another request for the same cheque will be rejected. The second e-cheque deposit request is rejected during clearance process at drawer's bank level. Hence, the proposed system is able to detect double spending of e-cheque.

4. Conclusion. This paper proposes a novel approach for transacting with e-cheque in banking to improve the clearance time and to reduce the manpower requirement in processing of cheque request. The approach is based on the blockchain technology and can be adopted by the current banking system with minimum integration effort. In order to achieve this, an efficient leader election and trust based consensus mechanism is proposed. On an average only 32.2% of nodes participate in the proposed trust based consensus mechanism and therefore message exchange per consensus process is much lesser as compared to traditional PoW approach thus making the system scalable. This reduces the communication overheads by using multicast instead of broadcast during consensus message exchange of messages. The time required to achieve the consensus for any new block is 25% lesser as compared to existing approaches such as PoW. The average time reduction for searching any cheque transaction details is more than 60% on an average aiding in faster clearance of the pending cheque transactions. Hence, the proposed e-cheque transaction framework is suitable to be deployed in real time banking and increase in number of transaction shall not degrade the performance of this system, making it scalable.

REFERENCES

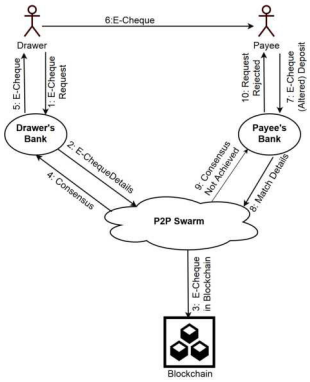


FIG. 3.8. *Detection of E-Cheque Alteration*

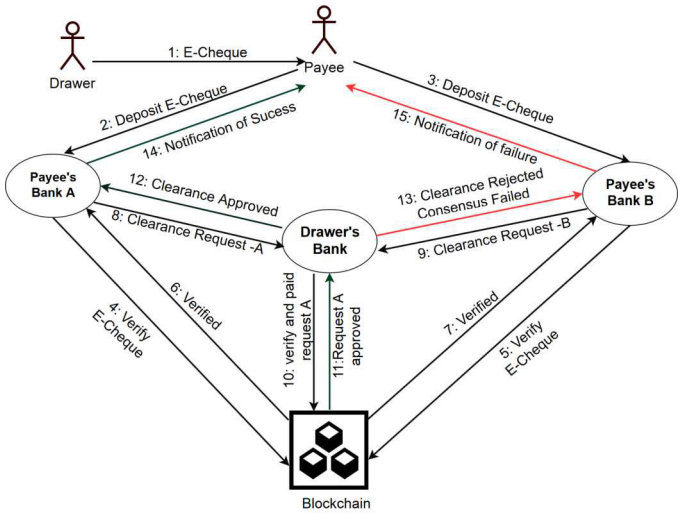


FIG. 3.9. *Handling threat of double spending of e-cheques*

[1] M. ANDERSON, *The electronic check architecture*, Financial Services Technology Consortium, 123 (1998).
[2] ANONYMOUS, *Introduction of indian banking system: Past and present senario*, http://shodhganga.inflibnet.ac.in/bitstream/10603/92717/9/, accessed on 2019-01-10.
[3] S. BANO, A. SONNINO, M. AL-BASSAM, S. AZOUVI, P. MCCORRY, S. MEIKLEJOHN, AND G. DANEZIS, *Consensus in the age of blockchains*, arXiv preprint arXiv:1711.03936, (2017).
[4] I. BARCLAYS, *Barclays says conducts first blockchain-based trade-finance deal*, https://reut.rs/2AQEG9w, accessed 2019-01-10.
[5] C.-C. CHANG, S.-C. CHANG, AND J.-S. LEE, *An on-line electronic check system with mutual authentication*, Computers & Electrical Engineering, 35 (2009), pp. 757–763.
[6] L. COCCO, A. PINNA, AND M. MARCHESI, *Banking on blockchain: Costs savings thanks to the blockchain technology*, Future Internet, 9 (2017), p. 25.
[7] I. EYAL, *Blockchain technology: Transforming libertarian cryptocurrency dreams to finance and banking realities*, Computer, 50 (2017), pp. 38–49.
[8] R. GJOMEMO, H. MALIK, N. SUMB, V. VENKATAKRISHNAN, AND R. ANSARI, *Digital check forgery attacks on client check truncation systems*, in International conference on financial cryptography and data security, Springer, 2014, pp. 3–20.
[9] Y. GUO AND C. LIANG, *Blockchain application and outlook in the banking industry*, Financial Innovation, 2 (2016), p. 24.
[10] INSURCHAIN, *Insurchain: A decentralized insurance blockchain ecosystem*, https://github.com/InsurChain/whitepaper/blob/master/en/whitepaper-en.md, accessed 2019-01-10.
[11] R. JAYADEVAN, S. R. KOLHE, P. M. PATIL, AND U. PAL, *Automatic processing of handwritten bank cheque images: a survey*, International Journal on Document Analysis and Recognition (IJDAR), 15 (2012), pp. 267–296.
[12] S. KING AND S. NADAL, *Ppcoin: Peer-to-peer crypto-currency with proof-of-stake*, self-published paper, August, 19 (2012).
[13] J. LIEBENAU AND S. ELALUF-CALDERWOOD, *Blockchain innovation beyond bitcoin and banking*, (2016).
[14] S. NAKAMOTO, *Bitcoin: A peer-to-peer electronic cash system*, (2008).

[15] Q. K. Nguyen, *Blockchain-a financial technology for future sustainable development*, in Green Technology and Sustainable Development (GTSD), International Conference on, IEEE, 2016, pp. 51–54.
[16] M. Rajender and R. Pal, *Detection of manipulated cheque images in cheque truncation system using mismatch in pixels*, in Business and Information Management (ICBIM), 2014 2nd International Conference on, IEEE, 2014, pp. 30–35.
[17] Santander, *Santander launches the first real-time trades in spain using we.trade, a blockchain platform that helps companies go international*, https://bit.ly/2Fw2pj7, accessed 2019-01-10.
[18] D. Schwartz, N. Youngs, A. Britto, et al., *The ripple protocol consensus algorithm*, Ripple Labs Inc White Paper, 5 (2014).
[19] K. Singh, N. Singh, and D. S. Kushwaha, *An interoperable and secure e-wallet architecture based on digital ledger technology using blockchain*, in 2018 International Conference on Computing, Power and Communication Technologies (GUCON), IEEE, 2018, pp. 165–169.
[20] N. Singh and M. Vardhan, *Distributed ledger technology based property transaction system with support for iot devices*, International Journal of Cloud Applications and Computing (IJCAC), 9 (2019), pp. 60–78.
[21] starbase, *Support innovative projects with star*, https://starbase.co/star?lang = en, accessed 2019-01-10.
[22] F. Tschorsch and B. Scheuermann, *Bitcoin and beyond: A technical survey on decentralized digital currencies*, IEEE Communications Surveys & Tutorials, 18 (2016), pp. 2084–2123.

Edited by: Dana Petcu
Received: January 18, 2019
Accepted: August 16, 2019

DOI 10.12694/scpe.v20i3.1548
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 3, pp. 527–540. http://www.scpe.org

OPTIMIZED SCHEDULING APPROACH FOR SCIENTIFIC APPLICATIONS BASED ON CLUSTERING IN CLOUD COMPUTING ENVIRONMENT

WALID KADRI* AND BELABBAS YAGOUBI†

Abstract. Cloud Computing becomes an important technology for the supplying of resources that can be used to execute large-scale scientific applications. It also provides lower-cost Computing resources access with personalized configurations. The user does not have to invest much in the acquisition and management of hardware such as storage, computing, databases, and networking, usually issued by the cloud provider. Users typically pay only for cloud services they use. Scientific applications usually represented as Directed Acyclic Graphs (DAGs) are an important class of applications that lead to challenging problems for resource management in distributed computing. With the advent of Cloud Computing, particularly the Infrastructure as a Service (IaaS) offers on-demand virtual machines executing multiple tasks. These tasks consist of a large number of DAGs requiring elaborated scheduling and resource provisioning policies. In goal of optimization and fault tolerance, DAGs applications are generally partitioned into multiple parallel DAGs using clustering algorithm and assigned to Virtual Machine (VM). In this work, we investigate through simulation, the impact of clustering for both provisioning and scheduling policies in the total makespan and financial costs for execution of user's application. We implemented four scheduling policies well-known in distributed computing systems, and adapted clustering algorithm to our resource management policy that leases and destroys dynamically VMs. We show that dynamic resources management policy can achieve better performance in term of makespan and budget cost than static management policies when partitioning workflow applications into grouped tasks before mapping them on virtual machines (VMs). The execution time and budget cost can be considerably reduced by managing efficiently VMs in order to maximise resources utilization and reduce the number of under-loaded VMs.

Key words: Cloud Computing, Dependent Tasks, Scientific Applications, Workflows, Directed Acyclic Graph, Resource Management, Task Scheduling, Virtual Machine, Clustering, CloudSim Simulator.

AMS subject classifications. 68M14, 68M20

1. Introduction.

Cloud Computing plays an increasingly important role in domains that are closely related with the web, offering a wide variety of services (i.e computing, data storage). These services can be accessed at any time and from any location via a high speed internet access. More specifically, in the domain of scientific applications which are very complex due to the nature of their tasks, and very expensive to schedule and execute them in parallel machines.

Due to fact that resource allocation is an NP-complete problem [13], an optimal assignment for tasks is impossible to realize. Finding an optimal scheduling becomes even more complex when tasks are dependent. It is due to the precedence relation between tasks; For example, when $T_1 \rightarrow T_2$, it means that T_2 cannot start until T_1 has been processed.

Scientific applications can be a set of dependent tasks with different granularity and different level. Many known companies in the Information Technology world offer a wide range of services with on demand payment (the user only pays for what he consumes). These services range from the provision of virtual machine instances such as AmazonEC2 [3], to parallel computing services whose main providers are Google and Yahoo.

Due to the importance of work applications, several research projects have been conducted to develop workflow management systems with scheduling algorithms. The projects: Condor Dagman [27], Gridbus toolkit [22], Iceni [26], Pegasus [11], and so forth, are designed for Grids, whereas cloudbus toolkit [23], SwinDeW-C [30], VGrADS [24], and so forth, are developed for Clouds. These systems can be viewed as a type of platform service facilitating the automation of scientific and commercial applications on the Grid and Cloud by masking their orchestrations and executions.

In this paper, we focus on the way how to manage effectively dependent tasks applications on Cloud environments and measure the impact of clustering algorithm in scheduling. We address the interaction between scheduling and resource management combined with clustering of DAGs and their impact on costs and run-time performance. We also make a comparison between generic approaches by selecting those likely to be the best.

*LIO Laboratory, Deptartment of Computer Science, University of Oran 1, Ahmed Ben Bella, Oran, Algeria (kadri.walid@univ-oran1.dz)

†LIO Laboratory, Deptartment of Computer Science, University of Oran 1, Ahmed Ben Bella, Oran, Algeria (b.yagoubi@univ-oran1.dz)

The paper is organized as follows: the first section introduces cloud computing and includes definitions, architectures, deployment and services models. The second section presents the related work to the issue addressed in this paper, namely DAG scheduling and resources management. Through the definition of objective functions of scheduling, we determine the criteria studied and present tasks scheduling algorithms implemented in our work for results comparison. The third section illustrates in detail the type of Scientific Application Models tested in our work with explaining each of them. Section 4 shows the implementation and simulation parameters like hardware configuration and performance metrics. We also define our simulation environment and a Cloud Computing simulator used and implemented algorithms are also detailed in our approach and the different scenarios simulated. We represent the obtained results into graph and analyze them in terms of performance metrics as time execution, complexity and financial costs. We conclude our paper with a conclusion and some future works.

2. Survey of Scheduling Strategies on Cloud Environment. This section survey all developed algorithm in scheduling of applications in cloud environments like scientific workflows and BoT (Bag of Tasks) or independent tasks. In particular, it focused on techniques considering applications modeled as DAGs and the resource model offered by public cloud providers. It presents a taxonomy of the existing scheduling algorithms on Cloud Computing.

The authors on [18] propose a mathematical model that optimizes the cost of scheduling workflows with a time constraint in a multi-cloud environment where each provider proposes a number of heterogeneous virtual machines or a global storage service for intermediate data files. Their method formulate the scheduling problem as Mixed Integer Program (MIP) and propose an overall optimization of placement and data sharing. Two types of workflow granularity are presented, coarse granularity in which tasks must be run for one hour, and the other for fine granularity workflow with shorter tasks and smaller deadline.

In the paper [20], the authors developed a planning service for middleware in the cloud, allowed optimal use of resources in terms of the total number of resources used in a defined interval given by user. They have proved the feasibility of their solution with taking into account dependencies between services.

The authors on [6] propose a dynamic re-configurable framework for the deployment of scientific workflows in the Cloud (called DR-SWDF). The user customize the workflow deployment process with a given set of objectives and constraints. The DR-SWDF framework offers a dynamic clustering of workflows based on K-means algorithm in order to identify the most convenient techniques or algorithms can be applied for their scheduling and deployment.

The work presented on [14] evaluate the impact performance of HPC applications on Microsoft Azure cloud platform using NAS parallel benchmarks. These benchmarks are used to test the communication performance. They have measured the speedup and MOPS for different scheduling allocation strategies and the results show that allocating one process per instance achieves higher scalability in term of the cost. The results are compared with the same configuration in Amazon platform and found that Azure platform has better shared-memory communication performance than Amazon platform. However, their work was designed for HPC Cloud and not for Cloud computing environment.

Authors in [12] developed the Multi-objective Heterogeneous Earliest Finish Time (MOHEFT) algorithm which as is an extension of the well-known DAG scheduling algorithm HEFT [29]. A set of pareto based solutions are computed from which users can select the suited one. The proposed algorithm builds intermediate workflow schedules, or solutions, in parallel in each step, instead of a single one as is done by HEFT. A crowding distance is used as metric between tasks characterized by dominance relationships in goal of finding solutions.

SABA [17] is a scheduling workflows algorithm proposed in a multi-cloud environment. The authors define the concept of immovable datasets which are restricted to a single data center and cannot be migrated or replicated due to security or cost concerns and movable datasets with no security restrictions and can be replicated. Their approach only considers the CPU capacity of VMs to estimate runtimes and features such as I/O, bandwidth, and memory capacity.

The PSO-based algorithm developed in [19] concerns a cost minimization, and a deadline-constrained algorithm that ensures dynamic provisioning and heterogeneity of computing resources. The authors have modeled the resource provisioning and scheduling as a Particle Swarm Optimization (PSO) problem. The output of the algorithm is a near-optimal schedule when determining the number and types of VMs to use, as well as their

leasing periods and the task to resource mapping. The results of the approach show that the computational overhead increases rapidly with the number of tasks in the workflow and the configuration of VMs provided to the user.

The IaaS Cloud Partial Critical Path (IC-PCP) algorithm [1] aims to minimize execution costs with a deadline constraint. The algorithm begins by recursively identifying a set of tasks with partial critical paths (PCP) associated to each exit node (an exit node is a node without child tasks). The tasks in each path are then placed on the same virtual machine with an instance already leased to meet the latest makespan requirements. In the case where the placement fails, the tasks are assigned to a least expensive, newly instantiated virtual machine that can execute the tasks. The process is repeated until the workflow is fully executed.

Authors in [7] adopt the main heuristic of IC-PCP of [29] and [1] by identifying partial critical paths and assigning their tasks to the same VM. They propose the Enhanced IC-PCP with Replication (EIPR) algorithm for scheduling and allocation that uses the idle time of allocated virtual machines and a budget exceeding to replicate tasks in order to minimize performance variations and deadline of applications. The algorithm starts by determining the number and type of virtual machines to use, the order and placement of tasks on these resources, and then determines the start time and end time of virtual machines.

Authors in [28] focus on their work on SLA when scheduling Workflow by implementing a SaaS provider offering a workflow execution service. They consider two types of SLA contracts that can be used to lease VMs from IaaS providers: static and subscription based. Specifically they consider offers configuration of Amazon EC2 [3], namely on-demand and reserved instances. In their proposed model, the SaaS provider has a pool of reserved instances that are used to execute workflows before a user-defined deadline. However, if the reserved instance infrastructure is not enough to satisfy the deadline, then on-demand instances are acquired and used to meet the workflow's requirements. Their algorithm is capable of selecting the best-suited IaaS provider as well as the VMs required to guarantee the QoS parameters.

The authors of [31] propose a scheduling framework that takes into account the dynamic nature of cloud environments in terms of performance and pricing. It is based on the same resource model as Amazon EC2 and takes Spot and on-demand instances into account. The main objective is to minimize the costs of executing workflows by providing a guarantee on probabilistic calculated deadlines based on the variability of resource performance and price of Spot instances. Spot instances aim to reduce infrastructure costs while on-demand instances ensure deadline compliance when spots instances are unable to complete tasks on time. This is achieved by generating a static hybrid instance configuration plan (a combination of spot and on-demand instances) for each task.

The authors of [25] implement various VM allocation and VM selection methods to effectively schedule the user tasks on the fog computing resources and try to find the best combination of task scheduling combination for the effective and optimized user data processing.

In the paper [16], authors propose hybrid balanced task clustering algorithm that uses the parameter of impact factor of workflows along with the structure of workflow. According to this technique, tasks can be considered for clustering either vertically or horizontally based on the value of the impact factor. This minimizes the system overheads and the makespan for execution of a workflow. A simulation based evaluation is performed on real workflows that shows the proposed algorithm is efficient in recommending clusters. It shows improvement of 5-10% in makespan time of workflow depending on the type of workflow used.

In the paper [5], Biswas and *al.* have proposed a Gravitational Search Algorithm (GSA) based workflow scheduling for heterogeneous computing systems. The proposed algorithm considers many objectives such as minimization of makespan, load-balancing, and energy-consumption. Their algorithm is designed to generate a valid execution sequence of tasks recursively with respecting the precedence relationship.

The authors of [2] present a non-dominance sort based Hybrid Particle Swarm Optimization (HPSO) algorithm to handle the workflow scheduling problem on Iaas clouds. The algorithm is a hybridation of their previous algorithm called Budget and Deadline constrained Heterogeneous Earliest Finish Time (BDHEFT) algorithm and multi-objective PSO. The HPSO heuristic tries to optimize makespan and cost under deadline and budget contraints and presents best solutions using pareto, and the final choice is done by user.

Kanagaraj and *Swamynathan* propose on [15] an effective resource provisioning and scheduling mechanism for workflow on cloud computing environment. The main idea is to determine the required number of VMs and

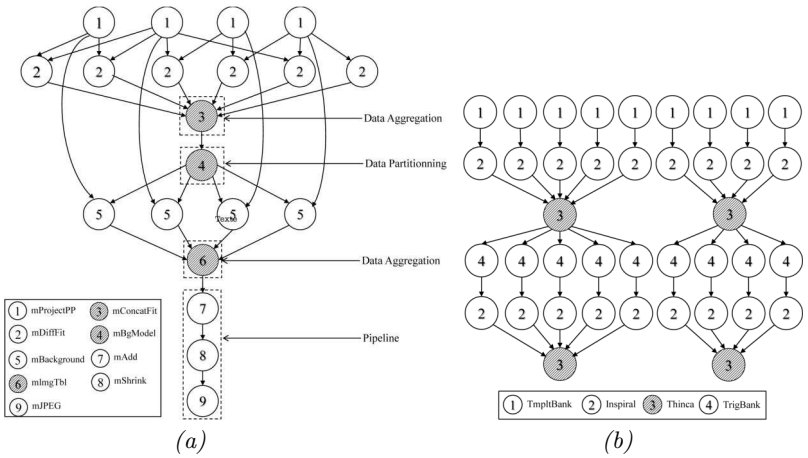


FIG. 3.1. *Montage DAG (a), and LIGO DAG (b).*

their configuration in a goal of optimizing data transfer between workflow tasks.

3. Scientific Applications Model (DAGs). In this section, we enumerate some of scientific applications (DAGs or also called Workflows) given by user and scheduled to VMs for execution using different assignment algorithms. To generate such scientific applications, we used a Pegasus Workflow Generator for DAX (Directed Acyclic Graph in XML) [21, 10] that provides a resource independent workflow description. It captures all the tasks that perform computation, the execution order of these tasks represented as edges in a DAG, and for each task the required inputs, expected outputs, and the arguments with which the task should be invoked. Pegasus provides simple, easy to use programmatic API's in Python, Java, and Perl for the DAX generation. We used the JAVA's API one for our work. The reason why we used a DAX generator is to avoid random generation of DAGs (nodes and links) which cannot be able to prove the effectiveness of our scheduling algorithm. The workflow applications used on our simulation are taken from [10] and described as follow:

3.1. Montage. This type of workflow application shown in Fig. 3.1 has been developed by the NASA/IPAC Infrared Science Archive and used to generate custom mosaics of the sky using images in the Flexible Image Transport System (FITS) format as input.

3.2. LIGO. Laser Interferometer Gravitational Wave Observatory (LIGO) scientific applications are used to search for gravitational wave as shown in Fig. 3.1.

3.3. CyberShake. CyberShake is a scientific application used in geology domain that calculates Probabilistic parameters for geographic sites. It identifies all ruptures and then calculates synthetic seismograms for each rupture variance (Fig. 3.2(a)).

3.4. Epigenomics. Epigenomics is a data-parallel scientific application used in Genetic domain. The Fig. 3.2.(b) shows this type workflow witch consists on Genetic data analyzed in the form of DNA sequence lanes. Each analyze can generate multiple lanes of DNA sequences and converted into a specific format. The mapping software can do one of two major tasks. The scientific application maps DNA sequences to the correct locations in a reference Genome.

3.5. SIPHT. Conducts a wide search for small untranslated RNAs (sRNAs) that regulates several processes such as secretion or virulence in bacteria. The Fig. 3.2.(c) shows a representation of Sipht DAG. The kingdom-wide prediction and annotation of sRNA encoding genes involves a variety of individual programs that are executed in the proper order using Pegasus.

4. Implementation and Simulation. This section presents the performance metrics used, different executed scenarios, the obtained results in the simulations, and their interpretations.

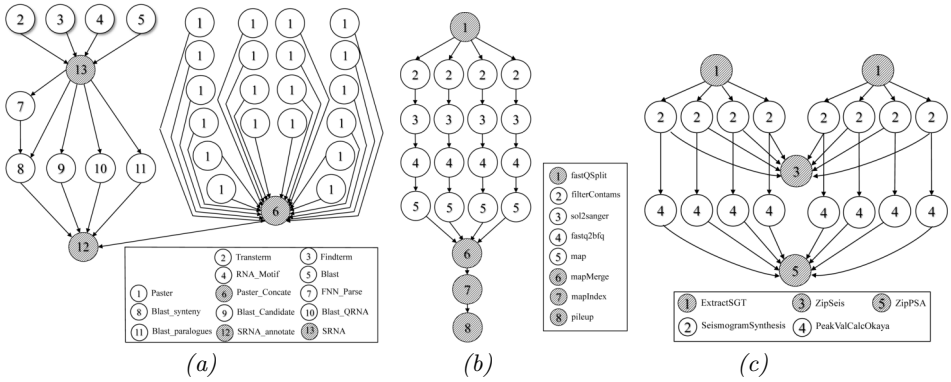


FIG. 3.2. *CyberShake DAG (a), and Epigenomics DAG (b) and Sipht DAG (c)*

4.1. Proposed Resources Management Approach. The algorithm 1 was designed after testing numbers of approaches for creating VMs. It allows to create dynamically virtual machines on demand and power-off the unused ones after a period of time. In fact, it is very difficult to provide a precise number of VMs that will be needed for the application, due to the nature of the jobs (DAGs) and their difference in term of execution time, and user's budget cost.

After testing the number of application, execution time, and finally the depth of each application (dependency levels), we have developed the algorithm described in Alg. 1. The main idea is to browse every level of dependence on the application, from the root to the leaves and increment the number of created VMs at every parallel jobs found, i.e after partitioning the graph into independent jobs as described in section 4.3.

Algorithm 1 VMs Management Algorithm

Require: *jobs*: Set of arrival jobs at t; t: current time;
function VMSCREATION(jobs)
  $VmList \leftarrow \emptyset$ ▷ List of VMs to return
  $Size \leftarrow$ ESTIMATESIZE($jobs$); $VmList \leftarrow$ REQUESTVMS($Size$) ▷ Send request to DataCenter
  return $VmList$
end function
function ESTIMATESIZE(jobs)
  $size \leftarrow 1$; $p_{previous} \leftarrow 0$ ▷ Depth of the previous job
  $depth_{max} \leftarrow jobs.getDepth()$ ▷ the maximal depth of jobs
  for $j \in jobs$ **do**
    $p_{Current} \leftarrow j.getDepth()$
    if $p_{Current} == p_{previous}$ **then** $size++$;
    end if
  end for
  return $size/depth_{max}$
end function

4.2. Creation and Submission of jobs. Workflows are generated from XML files. Generation of applications is as follows:
— A file is given as input. The file is interpreted from its XML structure and tasks are created.
— Jobs are then processed to add dependencies between tasks and creating a dependence matrix.
— Jobs are passed to the main Scheduler, which is responsible for dispatching jobs through the resource.
— Clustering process can be applied to jobs before scheduling.
The job creation and submission shown in Fig. 4.1 illustrate an example of the process from job submission by user until getting results feedback after determining the adequate job scheduling algorithm.

4.3. Clustering Algorithms. Clustering Algorithms (partitioning algorithms) consists on grouping different dependant tasks in the same cluster and executing them on the same virtual machine. A problem becomes complex when the number of clusters is larger than the number of the available virtual machines. It means that

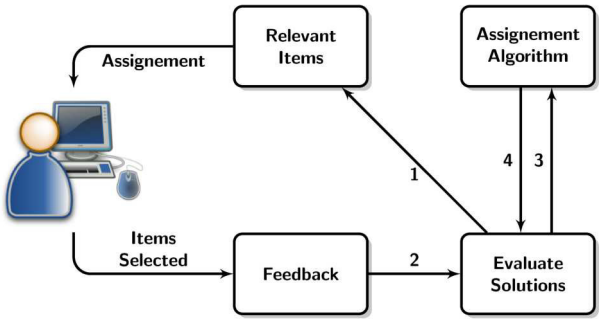


FIG. 4.1. *Creation and DAGs scheduling Process*

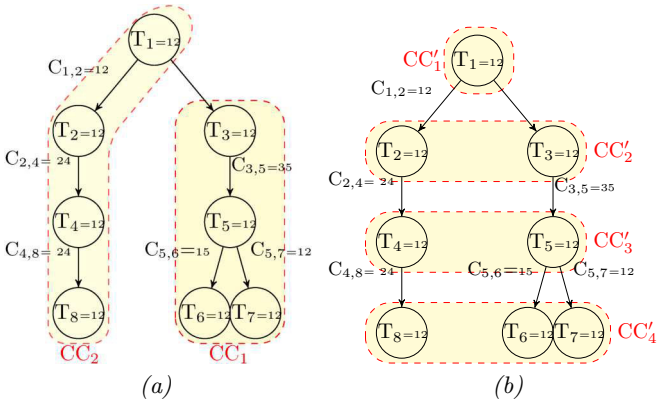


FIG. 4.2. *Vertical Clustering (a), and Level-based clustering (Horizontal Clustering) (b)*

the scheduling of several clusters onto the same virtual machines can increase considerably the overall execution time. We have to find the best mapping for grouped tasks on virtual machine according to its capability and its load. A load balancing algorithm is used in our case to achieve good performance.

In our work, we must determine a certain number of clusters which have a minimum makespan to be assigned into a virtual machine. We have used generic algorithms for tasks partitioning to decompose workflows (DAGs) submitted by user into related component (clusters) in order to schedule them to the same VM in order to minimize time response by reducing intercommunication costs between dependant tasks.

Two approaches have been adapted to our assignment approach: (i) **vertical clustering** described in algorithm 2 and Fig. 4.2 (a), and (ii) **horizontal clustering** described in algorithm 3 and Fig. 4.2 (b).

4.4. Simulation.

4.4.1. Configuration and execution parameters. The simulation was executed on a 64-bit MacOSx 12 work environment. The development environment used is Netbeans ver.7.0.1, all on a MacBook Pro machine with a i7 2.40GHz processor and 8GB of RAM. Simulated infrastructure is a Datacenter with 100 physical machines. Each physical machine can support 10 VMs, details on the configurations of the machines are given in the Table 4.1.

The simulator used is WorkflowSim [9] is an extension of CloudSim Simulator [8].

4.4.2. Performances Metrics. In this paragraph, we present the performance measures on which we have supported to first interpret the results and compare the different approaches. The two main performance measures are the overall implementation time and financial cost. These are conventional steps to test the effectiveness of scheduling algorithms and resource management.

Makespan . T_i being the end date of the job i

$$Makespan = \max T_i \tag{4.1}$$

Algorithm 2 Vertical Clustering Algorithm

Require: :$stack \leftarrow \emptyset$; $JobsList \leftarrow \emptyset$; $List_{temporary} \leftarrow \emptyset$

```
1: function VertiClustering(TasksList, dephtJob)                ▷ Level number of dependency
2:     if dephtJob > 0 then
3:         for t ∈ TasksList do
4:             if t.depth == 0 then Push(stack, t)               ▷ Stacking roots
5:             end if
6:         end for
7:         while stack ≠ ∅ do
8:             node ← Pop(stack)
9:             if !processed(node) then                          ▷ if the node have not been processed yet
10:                nbParent ← node.getNbParent()                 ▷ Number of previous tasks
11:                nbChildren ← node.getNbChildren()             ▷ Number of next tasks
12:                for cNode in node.getChildList() do
13:                    Push(stack,cNode)
14:                end for                                       ▷ Stacking all next tasks
15:                if nbParent! = 0 then                         ▷ if Root, Nothing to do
16:                    if nbParent > 1 then
17:                        if nbChildren > 1 ∨ nbChildren == 0 then
18:                            Liste_temporary.add(node)
19:                            addToJobs()
20:                        else                                  ▷ Only one next task
21:                            addToJobs(); Liste_temporary.add(node)
22:                        end if
23:                    end if
24:                    if nbChildren > 1 ∨ nbChildren == 0 then
25:                        addToJobs(); Liste_temporary.add(node)
26:                    end if
27:                end if
28:            else addToJobs()
29:            end if
30:            tag(node)                                         ▷ Tag the node as processed
31:        end while
32:    end if
33:    return JobList
34: end function
35: function addToJobs( )
36:    if Liste_temporary ≠ ∅ then
37:        JobList.add(CreateJob(List_temporary)); List_temporary ← ∅
38:    end if
39: end function
```

Table 4.1
Configuration of simulated machines

Configuration	physical Machine	VM
MIPS (Million Instructions per sec.)	2000	1000
RAM	4048 (MB)	2048 (MB)
PE (Processing Element)	4	2
Bandwidth	10000 (MB/S)	1000 (MB/S)
Space (storage)	1 (TB)	6 (GB)

$Makespan$ is simply the date of the last job to be done among all executed jobs.

Budget Cost. For the budget, the calculation model used is similar to a *a1.medium* machine price of *AmazonEC2 platform*[4], i.e periods of an hour for the rental of virtual machines. To simplify the interpretation of results, VMs costs have been reduced to $0.051\$/Hour$. Once a machine is created, an additional cost is added, rounding the cost another time.

Algorithm 3 Horizontal Clustering Algorithm

```
1: function HorizClustering(TasksList, ClusterSize)
2:     depht_max ← 0 ; JobsList, List_temporary ← ∅                ▷ Partitioned Jobs to return
3:     i ← 0; TasksList_temporary ← 0
4:     if ClusterSize > 0 then                                      ▷ Size of partition
5:         for Task t ∈ TasksList do
6:             if t.depth > depht_max then
7:                 depht_max ← t.depth
8:             end if
9:         end for
10:        while i < depht_max do                                   ▷ For every level of tasks graph
11:            for Task t ∈ TasksList do
12:                if t.depth == i then
13:                    Push(Liste_temporary, t)
14:                end if                                           ▷ Temporary list for every level of graph
15:            end for
16:            Shuffle(List_temporary)                              ▷ Random Permutation
17:            while List_temporary ≠ ∅ do                          ▷ All tasks of this level not yet gathered
18:                if ClusterSize < List_temporary.length then
19:                    for j ← 0 ; j < ClusterSize do               ▷ Group tasks as cluster tasks
20:                        Task t ← Pop(List_temporary); Push(TasksList_temporary, t)
21:                    end for
22:                    JobsList.add(CreateJob(TasksList_temporary))
23:                else
24:                    for j ← 0 ; Liste_temporary.length do
25:                        Tache t ← Pop(List_temporary); Push(TasksList_temporary, t)
26:                    end for
27:                    JobsList.add(CreateJob(TasksList_temporary))
28:                end if
29:                TasksList_temporary ← ∅
30:            end while
31:        end while
32:    end if
33:    return JobsList
34: end function
```

Cost/VM. The cost for the ith VM is :

$$Cost_{VM_i} = \frac{Time_{secondes}}{3600} * 0.051\$ \tag{4.2}$$

Overall cost. The overall cost is :

$$Cost = \sum_{i=1}^{NumberVM} Cost_{VM_i} \tag{4.3}$$

4.4.3. Implemented algorithms for comparison.

FCFS (First Come First Served). is the simplest scheduling algorithm that simply queues tasks in the order that they arrive in the ready queue. The tasks that comes first will be executed first and next tasks starts only after the previous gets fully executed. It provides efficient, error-free and simple process for scheduling by saving the VMs or resources in cloud computing.

Minimum Execution Time (MET). Minimum Execution Time (MET) algorithm determines the best predictable completion time for VMs for a given task and decide to assign it to the this resource without regarding to its availability. The MET algorithm can sometimes lead to high load imbalance since the assignment is not dependent on the availability of VMs.

MinMin. MinMin algorithm selects from a set of unscheduled tasks and determines for each task the minimum completion times all virtual machines. The task with generally minimum completion time is chosen and scheduled on the resultant virtual machine. The scheduled task is then removed from task queue and the process is repeated until the all unscheduled tasks are performed.

TABLE 4.2
Specification of simulated scenario and arrival time of of each workflow

Application	Arrival Time
Insipral 1000 Tasks	t=0ms
Epigenomics 460 Tasks	t=100s
LIGO 1000 Tasks	t=200s
CyberShake 3000 Tasks	t=250s
Montage 250 Tasks	t=350s
Sipht 1000 Tasks	t=380s

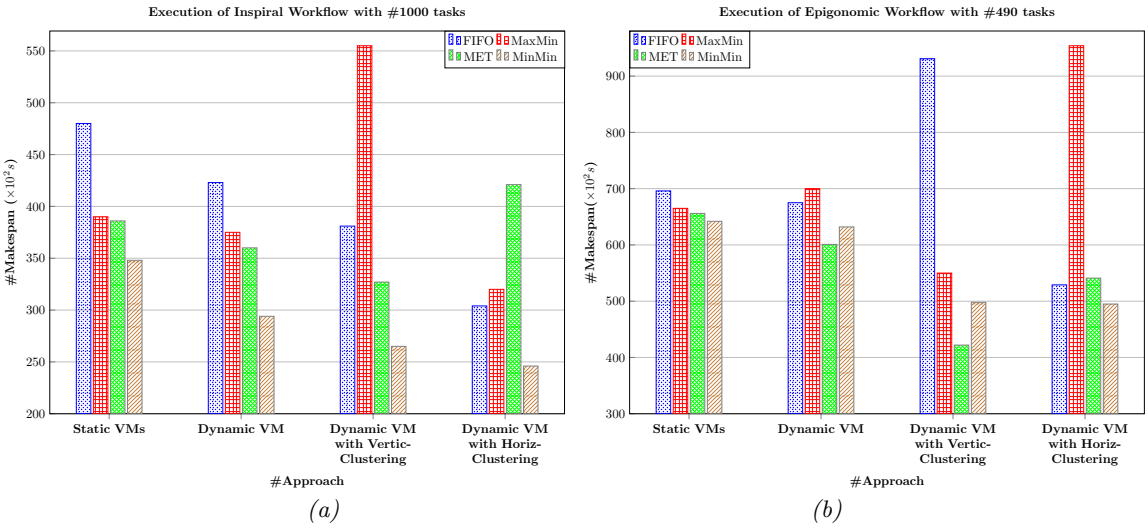


FIG. 5.1. *Makespan for Inspiral (a) and Epigenomic (b)*

MaxMin. Similar to the MinMin algorithm, it determins the completion times for each task on all virtual machines, the task with maximum completion time is scheduled on the consistent virtual machine in the case of MaxMin, and the process is the repeated until all the tasks are scheduled. MaxMin algorithm is usually employed in a situation where there are fewer longer and shorter tasks. Smaller makespan low degree of imbalance among virtual machines is guaranteed if more tasks are scheduled on machines that execute them earliest and fastest due to the predictable makespan and the real time workload evaluation of VMs.

4.4.4. Simulation Scenario. The scenario realized aims to model applications in realistic way, and that, by highlighting several applications with deadlines arrivals between them. The Table 4.2 shows the generated applications in this scenario, with the number of tasks included on each of them. The number of jobs remains the same when the configuration is changed.

5. Simulation Results and Discussion. This section summarizes the findings and contributions made. we will illustrate some experimental results in order to demonstrate the effectiveness of our dynamic management and scheduling resource of the different workflows described on section 3 and measuring the impact of clustering on them. As we can see in Figs. 5.1 (a), 5.1 (b), 5.2 (a), 5.2 (b), 5.3 (a), 5.3 (b) show respectively the makespan time scheduling of Inspiral, Epigenomics, LIGO, Cybershake, Montage, and Sipht scientific workflows into VMs according to implemented scheduling algorithms. We have tested 4 generic algorithms described in section 4.4.3 FIFO, MET MaxMin, and MinMin with an initial number of VMs fixed to 50, and have combined each one with static management VM, our Dynamic Management VM approach, our Dynamic VM management approach combined vertical Clustering, and finally with our Dynamic VM management approach combined horizontal Clustering. We have also measured in Fig. 5.4, the cost of running the user's application described in Table 4.2 and the average resource occupation in terms of CPU and Memory (see Fig. 5.5 (a) and Fig. 5.5(b)).

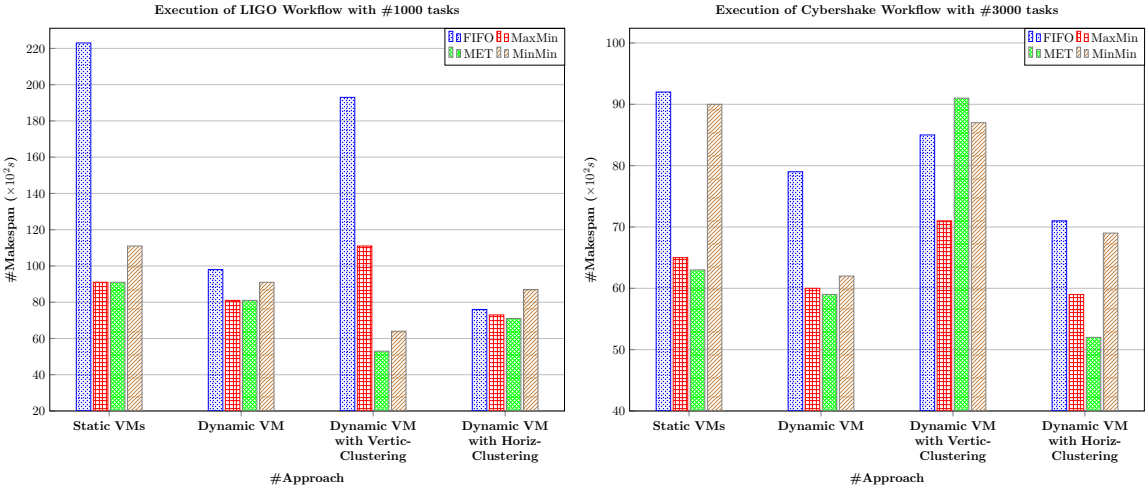


FIG. 5.2. *Makespan for LIGO (a) and Cybershake (b)*

The execution cost is shown in Fig. 5.6.

Figs. 5.1 (a) and 5.1 (b) show the maskespan in second of respectively the inspiral workflow with 1000 tasks and epigenpmics worflows with 460 tasks. From these results it is clear that the dynamic VM management with horizontal clustering gives better results than the other ones expects of MaxMin in vertical clustering (see Fig. 5.1 (a)) and horizontal clustering (see Fig. 5.1(b)) which give the worst makespan. This is due to the nature of Inspiral and Epigenomic worflows. The first one didn't support the vertical clustering due to the number of level of the DAG, and the second didn't support the horizontal clustering due to its number of nodes dependencies at the same level. In the Fig. 5.2 (a), the Ligo worfklow with 1000 tasks is performing very good results using our dynamic VM management approach comparing with the static one. The same analysis is given for Fig. 5.2 (b) which illustrate the cybershake workflow with 3000 tasks. The obtained results in Figs. 5.2 (a) and 5.2 (b) show that the makespan is better and the makespan decreases considerably from $230 \times 10^2(second)$ to $75 \times 10^2(second)$ for FIFO and from 90×10^2 to 72×10^2 for MaxMin when using Horizontal clustering with our management approach (see Fig. 5.2 (a)).

The Fig. 5.2 (b) globally gives good results with dynamic VM management. But we notice that when applying vertical clustering, MET gives the worst makespan. This is due to the vertical partitioning of the cybershake workflow into related jobs, this increase communication cost of the different clusters of this latter and as results, the makespan increase when waiting dependency results from jobs. It is worth discussing these interesting facts revealed by this results and can say that partitioning workflows didn't always give best makespan, it depends on the complexity of the workflow. In Fig. 5.3 (b) our dynamic management VM didn't perform good results with the Fifo algorithm, because of we didn't apply clustering, and when creating VM dynamically, we can disable non-allowed VM, and enable it when needed. This operation take an additional time which is non negligible and increase the makespan of the user's application.

We can also say that the adequate clustering for Sipht is the vertical one for MaxMin, MinMin end MET scheduling algorithm for our dynamic management algorithm, and the horizontal clustering for FIFO as shown in Fig. 5.3 (b). For the makespan of Ligo workflow, better is to use the dynamic VM management with vertical clustering for MET and MinMin, and the dynamic VM management with horizontal for FIFO and MaxMin scheduling algorithm.

Extensive results are illustrated in Fig. 5.4 carried out show that our dynamic management of VMs method reduce the user's budget cost for execution of its application. We can see that the cost of running the application (user's scenario described bellow) decrease considerably from approximately 5.5$ to 1.23$ when using dynamic VM management. Because of shutting down unused VMs and reducing makespan of user application.

However, in some cases, it's better using vertical than horizontal clustering when using complicated workflow

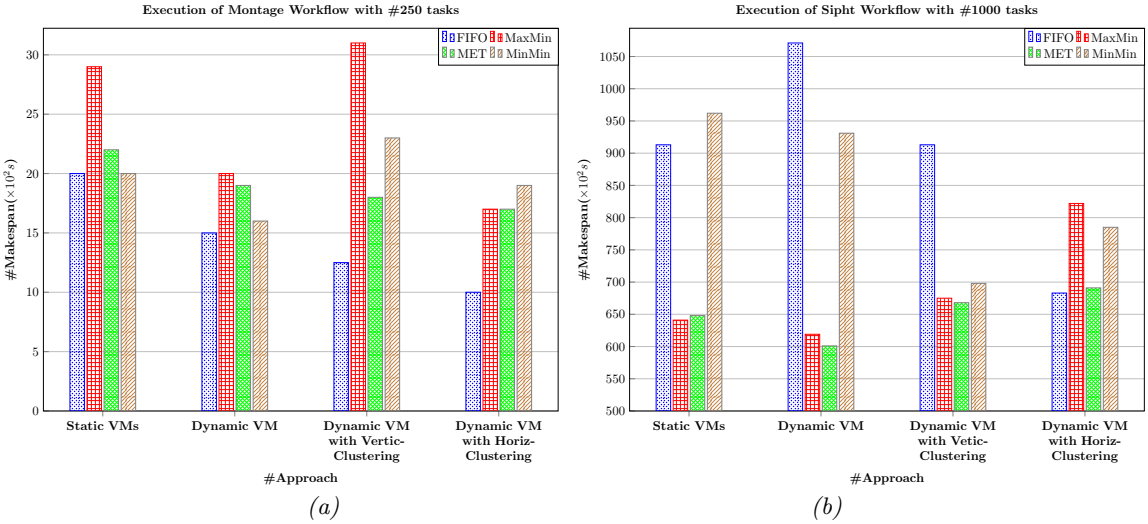


FIG. 5.3. *Makespan for Montage (a) and Sipht (b)*

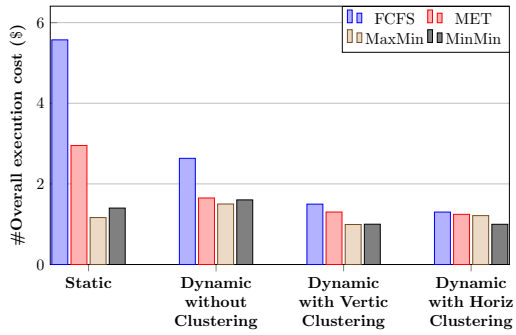


FIG. 5.4. *Execution Costs($) for the simulated scenario*

with a lot of dependencies such as LIGO and Sipht.

However, applying any clustering yields a lower cost as shown in Fig. 5.4. This can be explained by the effect of grouping tasks and mapping them into the same virtual machines. Indeed, the number of VMs created is more concise, and the VMs are used more efficiently. To conclude, we can say that the FCFS gives a higher costs than MET, MaxMin and MinMin algorithms which have slightly different costs. The obtained results in different simulation scenarios let us to think about proposing an adaptation of the vertical and horizontal clustering algorithm as a perspective to further reduce the cost of execution.

The Fig. 5.5 measures according to simulation time the number of VMs allowed for scheduling scenario 1 for both static (see Fig. 5.5 (a)) and dynamic (see Fig. 5.5 (b)) VM allocation. The dynamic VM allocation performs well, giving the minimum number of VM. It leads us to reduce the execution cost without increasing makespan. For the No clustering of DAGs, the number of VM converge to 36 VMs for static approach when this latter decrease to 26 VMs for our dynamic resource management. We can say that best results are obtained with the vertical clustering which is 20 VMs for the static resource management and 6 VMs the dynamic resource management. The peak of VM number (50 VM) in the graph is the started number of VM created initially is fixed in our simulation to 50 VMs.

The Horizontal clustering in Figs. 5.5 (a) and 5.5 (b) give medium results for the overall application, This suggests that the number of tasks of different workflows composing the user's application.

We describe the results of resource occupation graph, which is shown in Fig. 5.6. We understand why the application's execution cost is so high in Fig. 5.3 (a) when using static management approach. It demonstrates

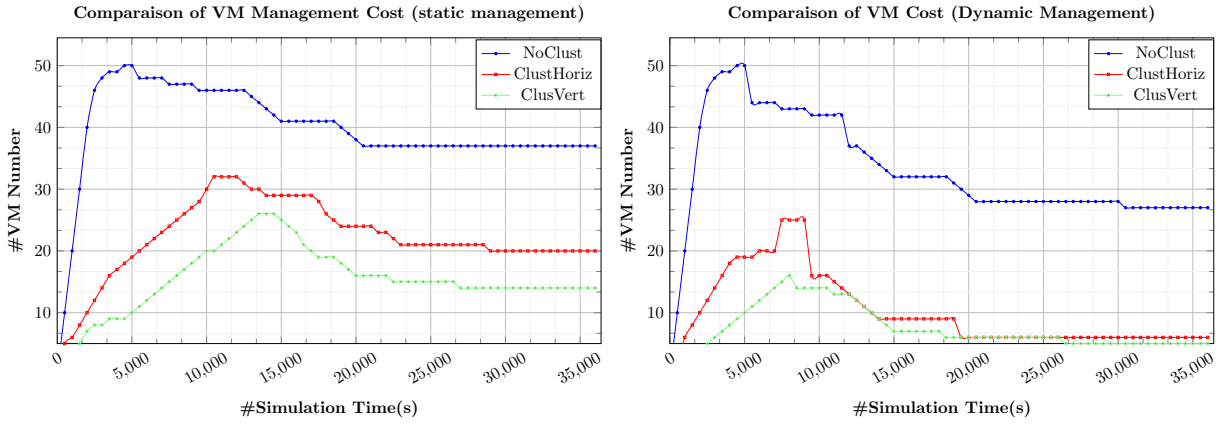


FIG. 5.5. *VMs management costs and clustering impact (Static (a) vs Dynamic (b))*

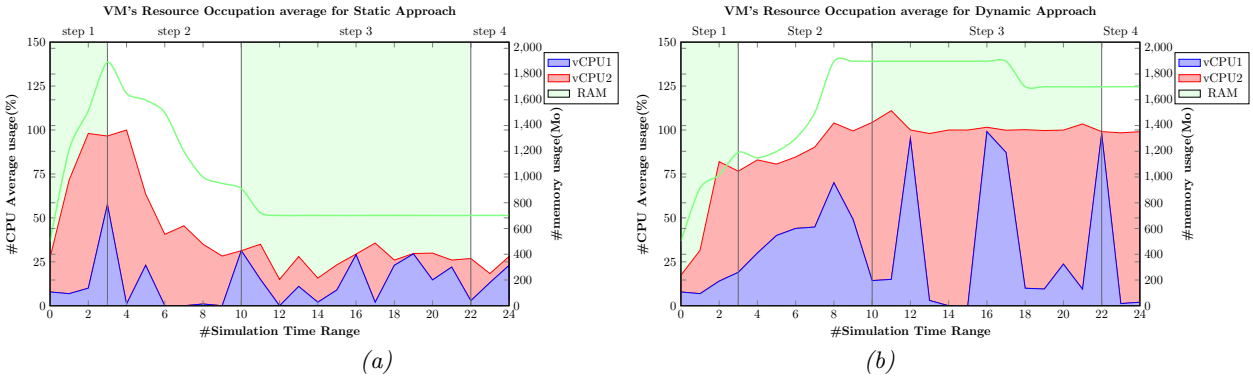


FIG. 5.6. *Resource occupation average of VM (Static (a) vs Dynamic (b))*

two things:
– First, with static management VMs, the resource utilization is not maximized and the VMs are running jobs but not effectively, and we notice that CPU average not exceed 35% the most of time. The same thing for the memory, it not exceeds 600MB from the hole 2048MB allowed for one Virtual Machine (see Fig. 5.6 (a)).
– Second, by reducing the number of VMs and maximizing the utilization of resource using our dynamic approach like showed in Fig. 5.6 where CPU average occupation is globally 100% and memory approaching the max capacity allowed (1400MB), we can reduce considerably the budget cost of running complex scientific application submitted by user.

From these results it is clearly proved that dynamic management approach reduce considerably the makespan and performs better results on most of cases. It depends on the complexity of scientific workflow submitted by the user, clustering can impact positively on performance metrics. But our approach has some limits and depends on partitioning the graph which can lead to higher makespan if the scientific workflow is very complex.

But globally, our approach leads to good results, even if the improvement is negligible in some cases.

6. Conclusion and Future Work. As big are the opportunities that cloud can offer, scheduling complex scientific application (workflows) and resource allocation plays a critical role on the Cloud. In fact give the user the ability to allocate the resources it needs is to give the control over the resources, so that he can manage by himself.

A proper dynamic scheduling mechanism will aid the user to reduce the cost and time of workflow execution. An optimized dynamic scheduling workflow proposed in this paper, analyses the structure of workflows and suggests an optimized resource provisioning approach after partitioning dependent tasks on related com-

ponent. This helps the users to allocate optimum number of resources with the required configuration reducing considerably execution cost and makespan.

In this case, we have addressed the problem of an optimized resource management and scheduling approach of tasks and their influence on the performance and cost execution of complex scientific applications running on IaaS Cloud. The initial objectives were:

— Propose a optimized scheduling approach and resource management policy for an IaaS cloud.

— Implement and study the impact on performance metrics of our approach with other scheduling policies and workflows clustering.

— Finding best combinations of scheduling and clustering depending on nature of workflows applications.

Four generic task scheduling algorithms were implemented and combined with two partitioning approaches for workflows. These algorithms were compared with our dynamic VM management policy that manage dynamically virtual machines according to their workload balance, to the nature and size of workflows.

Clustering the workflows before scheduling them with our dynamic management policy of VMs seems to be very effective for complex DAGs applications we tested. The MaxMin and MinMin algorithms which start with the largest (respectively, smaller) jobs and scheduling them on best resources, were slightly better on the FCFS (first come first served) and MET approaches, the first placing the jobs according to their arrival order by relegating the others in the queue, while the second takes the logic of the first without queuing them.

The simulation outcomes express that dynamic scheduling approach algorithm increases the utilization of resource and reduces the response time for workflow scheduling. The obtained results are quite encouraging and many perspective still open for our future work. We can mention some of these future directions:

— Integrating new tolerance failure mechanism of computing resources and migration techniques of VMs them into another VMs with saving its instance.

— Considering data storage for scientific applications that can affect considerably performance and execution cost.

REFERENCES

[1] S. Abrishami, M. Naghibzadeh, and D. H. Epema, *Deadline-constrained Workflow Scheduling Algorithms for Infrastructure As a Service Clouds*, Future Gener. Comput. Syst., 29 (2013), pp. 158–169, 10.1016/j.future.2012.05.004.

[2] V. Amandeep and K. Sakshi, *A Hybrid Multi-Objective Particle Swarm Optimization For Scientific Workflow Scheduling*, Parallel Computing, 62:C (2017), pp. 1–19, 0.1016/j.parco.2017.01.002.

[3] *Amazon EC2*, September (2017), https://aws.amazon.com/ec2.

[4] *Amazon EC2 Pricing*, March (2019), https://aws.amazon.com/fr/ec2/pricing/on-demand/.

[5] T. Biswas, P. Kuila, A. K. Ray, and M. Sarkar, *Gravitational Search Algorithm Based Novel Workflow Scheduling for Heterogeneous Computing Systems*, journal of Simulation Modelling Practice and Theory, 96 (2019), pp. 101932, 10.1016/j.simpat.2019.10193.

[6] K. Bousselmi, Z. Brahmi, and M. Mohsen Gammoudi, *DR-SWDF: A Dynamically Reconfigurable Framework for Scientific Workflows Deployment in the Cloud*, j-SCPE., 18:2(2017), pp. 177–193.

[7] R. N. Calheiros and R. Buyya, *Meeting Deadlines of Scientific Workflows in Public Clouds with Tasks Replication*, IEEE Trans. Parallel Distrib. Syst., 25 (2014), pp. 1787–1796, 10.1109/TPDS.2013.238.

[8] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, *Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms*, Softw. Pract. Exper., 42 (2011), pp. 23–50.

[9] W. Chen and E. Deelman, *WorkflowSim: A Toolkit for Simulating Scientific Workflows in Distributed Environments*, in Proceedings of the 2012 IEEE 8th International Conference on E-Science (e-Science), Washington, DC, USA, 2012, IEEE Computer Society, pp. 1–8, 10.1109/eScience.2012.6404430.

[10] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. F. da Silva, M. Livny, and K. Wenger, *Pegasus, a Workflow Management System for Science Automation*, Future Gener. Comput. Syst., 46 (2015), pp. 17–35, 10.1016/j.future.2014.10.008.

[11] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.-H. Su, K. Vahi, M. Livny, *Pegasus: Mapping Scientific Workflows onto the Grid*, in Grid Computing, Second European Across Grids Conference, AxGrids 2004, Nicosia, Cyprus, January 28-30, 2004, Revised Papers, 2004, pp. 11–20, 10.1007/978-3-540-28642-4_2.

[12] H. M. Fard, R. Prodan, J. J. D. Barrionuevo, and T. Fahringer, *A Multi-objective Approach for Workflow Scheduling in Heterogeneous Environments*, in Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid 2012), Washington, DC, USA, 2012, IEEE Computer Society, pp. 300–309, 10.1109/CCGrid.2012.114.

[13] M. R. Garey,and D. S. Johnson, *Using NP-Completeness to Analyze Problems*, in Computers and Intractability A guide to the theory of NP-completeness., 29 (2002), pp. 34–48.

[14] H. A. Hassan, A. I. Maiyza , and W. M. Sheta,*Impact of Process Allocation Strategies in High Performance Cloud Computing on Azure Platform*, j-SCPE., 18:2(2017), pp. 161–176, https://www.scpe.org/index.php/scpe/article/view/1288.
[15] K. Kanagaraj and S. Swamynathan, *Structure aware resource estimation for effective scheduling and execution of data intensive workflows in cloud*, Future Gener. Comput. Syst, 79 (2018), pp. 878–891.
[16] A. Kaur, P. Gupta, and M. Singh, *Hybrid Balanced Task Clustering Algorithm For Scientific Workflows in Cloud Cmputing*, j-SCPE, 20:2 (2019), pp. 237-258, 10.12694/scpe.v20i2.1515.
[17] Z. Lingfang, V. Bharadwaj, and L. Xiaorong, *SABA: A security-aware and budget-aware workflow scheduling strategy in clouds*, J. Parallel Distrib. Comput., 75 (2015), pp. 141–151, 10.1016/j.jpdc.2014.09.002.
[18] M. Malawski, K. Figiela, M. Bubak, E. Deelman, and J. Nabrzyski, *Scheduling Multilevel Deadline-constrained Scientific Workflows on Clouds Based on Cost Optimization*, Sci. Program., 2015 (2015), pp. 5:5–5:5, 10.1155/2015/680271.
[19] M. Masdari, F. Salehi, M. Jalali, and M. Bidaki, *A Survey of PSO-Based Scheduling Algorithms in Cloud Computing*, J. Netw. Syst. Manage., 25 (2017), pp. 122–158, 10.1007/s10922-016-9385-9.
[20] F. Nizamic, V. Degeler, and R. Groenboom, *Policy-Based Scheduling of Cloud Services*, j-SCPE., 13:3(2012), pp. 187–199.
[21] Pegasus, *Workflow Generator*, 2018.
[22] B. Rajkumar and V. Srikumar, *The Gridbus Toolkit for Service Oriented Grid and Utility Computing: An Overview and Status Report*, CoRR, cs.DC/0404027 (2004).
[23] B. Rajkumar, P. Suraj, and V. Christian, *Cloudbus Toolkit for Market-Oriented Cloud Computing*, CoRR, abs/0910.1974 (2009).
[24] L. Ramakrishnan, C. Koelbel, Y.-S. Kee, R. Wolski, D. Nurmi, D. Gannon, G. Obertelli, A. YarKhan, A. Mandal, T. M. Huang, K. Thyagaraja, and D. Zagorodnov, *VGrADS: Enabling e-Science Workflows on Grids and Clouds with Fault Tolerance*, in Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, New York, NY, USA, 2009, ACM, pp. 47:1–47:12, 10.1145/1654059.1654107.
[25] S. P. Singh, A. Nayyar, H. Kaur, and A. Singla, *Dynamic Task Scheduling Using Balanced VM Allocation Policy For FOG Computing Platforms*, j-SCPE, 20:2 (2019), pp. 433-456.
[26] M. Stephen, Y. Laurie, A. Ali, N. Steven, and D. John, *Workflow Enactment in ICENI*, in In UK e-Science All Hands Meeting, Publishing Ltd, 2004, pp. 894–900.
[27] D. Thain, T. Tannenbaum, and M. Livny, *Condor and the Grid*, 2003, 10.1002/0470867167.ch11.
[28] A. L. G. Thiago, F. B. Luiz, and R. M. M. Edmundo, *Workflow scheduling for SaaS / PaaS cloud providers considering two SLA levels*, in 2012 IEEE Network Operations and Management Symposium, NOMS 2012, Maui, HI, USA, April 16-20, 2012, 2012, pp. 906–912, 10.1109/NOMS.2012.6212007.
[29] H. Topcuouglu, S. Hariri, and M. you Wu, *Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing*, IEEE Trans. Parallel Distrib. Syst., 13 (2002), pp. 260–274, 10.1109/71.993206.
[30] Y. Yun, L. Ke, C. Jinjun, L. Xiao, Y. Dong, and J. Hai, *An Algorithm in SwinDeW-C for Scheduling Transaction-Intensive Cost-Constrained Cloud Workflows*, in Fourth International Conference on e-Science, e-Science 2008, 7-12 December 2008, Indianapolis, IN, USA, 2008, pp. 374–375, 10.1109/eScience.2008.93.
[31] A. C. Zhou, B. He, and C. Liu, *Monetary Cost Optimizations for Hosting Workflow-as-a-Service in IaaS Clouds*, IEEE Trans. Cloud Comput., 4 (2016), pp. 34–48, 10.1109/TCC.2015.2404807.

Edited by: Dana Petcu
Received: April 8, 2019
Accepted: July 26, 2019

DOI 10.12694/scpe.v20i3.1556
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 3, pp. 541–562. http://www.scpe.org

E-DPSIW-FCA: ENERGY AWARE FCA-BASED DATA PLACEMENT STRATEGY FOR INTENSIVE WORKFLOW

RIHAB DEROUICHE*, ZAKI BRAHMI†, MOHAMMED MOHSEN GAMMOUDI ‡, AND SEBASTIÁN GARCÍA GALÁN §

Abstract. Intensive Workflows are composed of large number of complex tasks and require a large amount of data located in different Storage Computing Servers (*SC*). The data movement between *SC* causes high communication and data movement cost. In this paper, a data placement strategy based on Formal Concept Analysis approach (E-DPSIW-FCA) is proposed aiming to reduce the data movement, the consumed energy, and the workflow execution cost. FCA allows to group the maximum of data and tasks in an hierarchical structure called lattice concepts. These concepts are mapped to the appropriate *SC*. The navigation through the hierarchy of concepts is considered as a solution of the case when the data group size exceeds the *SC* storage capacity. The simulations results show that E-DPSIW-FCA can achieve better results than the K-means [4] and genetic algorithm [14] based approaches.

Key words: Data Placement, Intensive Workflow, Cloud Computing, FCA, Energy, Communication, Computing, Granularity, Network, Level

AMS subject classifications. 68M14, 68P20

1. Introduction.

Workflows are composed of a sequence of operations and declared as a work of person or a group [1]. They have been used in a number of scientific applications, which generate massive amount of data every day, such as astronomy [2] and bioinformatics [3]. These workflows are potentially intensive and comprise hundreds or thousands of complex tasks and big datasets which need to be stored and, make its processing very complex [4]. For instance, the Cybershake workflow, which is used to calculate probabilistic seismic hazard curves for several geographic sites in the southern California area, has an average execution time that can be up to 8 hours and 51 minutes for running 2 083 325 tasks and generates a huge amount of datasets [5] [6]. So, in order to process and store these huge amounts of workflow data, Cloud providers are deploying large number of Computing and Storage devices to address and satisfy the ever increasing user's requirements for more computing capacity, storage and memory. As well as that, they offer different Cloud storage resources including Amazon Web Services (AWS) which offers various kinds of cloud storage systems [7]. For example, Elastic Block Store (EBS) provides persistent block storage volumes for use with Amazon Elastic Compute Cloud (Amazon EC2) instances whereas the data are delivered as data block [8]. Amazon Elastic Compute Cloud (Amazon EC2) instances and the International Business Machines (IBM) offered a block storage with up to 12 TB in capacities [9].

The execution of a workflow task requires a massive volume of datasets, which are physically distributed and stored in multiple *SC*. Thus, data movement between *SC* will be inevitable and it would generate a significant data movement cost due to the difference in the location between the data processing task and the necessary dataset. Consequently, transfer large amounts of data between *SC* increases notably the consumed energy by the networking devices, communication links, hence the increasing of the workflow execution cost. Further, processing such a large size of moved data for executing workflows in the Cloud stands as a challenge [1]. According to [10], a major portion of the consumed energy by the data center is utilized to maintain interconnection links and network equipment operations. This consumed energy account for up to 50% of the total energy consumption of a data center [11] and it is caused mainly by switches, Local Network Area infrastructure (or LAN), routers, etc [12]. Reducing the energy consumption of a Data Center Networks (DCNs) is considered as an essential step for advancing energy efficiency in Cloud Computing paradigm [11]. It is considered hence as a challenge [1].

To address data placement problem, many works had been proposed such as [3] [4] [13] [25], etc. However, to the best of our knowledge, most existing data placement studies have considered storing data in data centers

*Faculty of Sciences of Tunis, RIADI-GDL Laboratory, Tunisia, (derouicherihab@gmail.com).
†Taibah University, KSA, RIADI-GDL Laboratory, Tunisia, (zakibrahmi@gmail.com).
‡ISAMM, University of Mannouba, RIADI-GDL Laboratory, Tunisia, (gammoudimomo@gmail.com).
§Higher Polytechnic School of Linares, University of Jéan, Spain, (sgalan@ujaen.es).

without taken into account the granularity of data centers used resources, and the network aspect of a data center infrastructure. More specifically, they did not cope with the consumed energy incurred by the networking devices, the communication links and by the computing devices during the data placement and movement. Thus, we are interested in proposing a strategy for data placement that considers the aspects cited previously. To resolve this data placement problem, it is important to manage effectively these datasets in order to minimize both the energy consumption and the total data movement cost efficiently during the workflow execution. This can be achieved by placing and distributing intelligently these datasets in order to reduce the consumed energy by computing, storage and communication devices as well as reducing the cost of using these devices.

In this paper, a novel Data Placement Strategy based on Formal Concept Analysis (E-DPSDIW-FCA) is proposed. Indeed, contrarily to others studies, aiming to group the maximum of dependent datasets, our approach operates at the granularity of a data center while considering its different levels of communication (routers, switches, etc.), so that, to provide an efficient energy.

Overall, our main contributions are summarized as follows:

1. Applied the FCA approach in a new field which is massive data placement. Indeed, we used the FCA approach to identify the tasks-datasets associations (Formal concepts) which indicate the dependency among datasets and tasks,
2. Proposed a data placement algorithm to distribute and place original data and tasks based on their dependencies in SC. For this end, as to save lattice computing time, firstly, we propose a K-means based algorithm that allows dynamically clustering the input scientific workflows in order to identify the most convenient algorithm to apply for the lattice concept computing. Second, we reduce the size of the initial formal context. Additionally, an heuristic is defined by exploiting lattice concept level, and this in order to avoid examining all concepts at the process of mapping datasets to SC.
3. Proposed a new data transfer cost model closely to real Cloud environment by using data slices.
4. Proposed a novel energy model to evaluate the energy consumption by communication, computing and storage devices,
5. Formulated a mono-objective mathematical optimization model for workflow data placement taking into account both the communication energy and the computing energy consumption, the data movement and the quality metrics of server as explicit decision criteria, and
6. Implemented our algorithm and evaluate its performances with some scientific intensive workflows.

The remainder of this paper is organized as follows: in the section 2, we perform the system modeling and the problem formulation. In section 3, we detail our proposed solution and justify our choice for FCA. Section 4 demonstrates the simulation results and the evaluation of our approach. Section 5 highlights the major related works addressing the data placement problem and the existing energy models strategies. Finally, we conclude our work and we give some perspectives.

2. System Modeling and Problem Formulation.

2.1. Workflow Modeling. A workflow W is modeled as: $W = (T, D^{\text{in}}, TS, DS)$. $T = \{t_i | i = 1, ..., n\}$ represents the set of workflow tasks with n the number of tasks. $D^{\text{in}} = \{d_i^{\text{in}} | i = 1, ..., m\}$ represents the amount of original datasets to be processed by each workflow task t_i and m is the number of datasets that are consumed by such workflow task as input datasets. Each dataset d_i^{in} has a size denoted as $size(d_i^{\text{in}})$. This size is defined in some pre-determined unit such as mega-bytes, gigabytes or tera-bytes. Each task t_i requires a set of data denoted as $D^{\text{in}}(t_i)$ from the set D^{in} to satisfy its execution. TS: $D^{\text{in}} \rightarrow T$ is a function dataset-task that returns the set of workflow tasks that consume such datasets as their inputs. DS: $T \rightarrow D^{\text{in}}$ is a function task-dataset that returns the set of datasets that are consumed by such task as its inputs.

2.2. Cloud Computing Environment Modeling. A Cloud Computing environment is a 3-tuple: $CC = (DCN, SC, Scap)$ where

(i) DCN represents the datacenter network architecture, that consists of a set of Computing and Networking devices. In addition, the computing devices are the servers (storage computing) denoted as SC and the networking devices are the network switches and routers denoted as SWs.

(ii) $SC = \{sc_i | i = 1, ..., s\}$ represents a set $'s'$ of storage computing servers from the Cloud environment. As mentioned above, SC may be virtual machines, dedicated storage sites, Amazon S3, Elastic Block Store,

etc. Each sc_i has a storage capacity denoted as $Cap(i)$.

The communication link among the SC is defined as the bandwidth demand or the traffic load between two SC sc_i and sc_j. Besides, this communication link is expressed as: $BW(sc_i, sc_j)$.

A computing server sc_i is defined by three quality metrics (M_i) used to evaluated its performance denoted as: $M_i = (C_{storage}, C_{processing}, C_{DataTransfer})$ [63].

Actually, our goal lying behind evaluating the performance of server sc_i consists in reducing the data transfer cost, the storage cost and the processing cost. Hence, the goal is to minimize the sum Q_i:

$$Q_i = C_{storage} + C_{processing} + C_{DataTransfer} \tag{2.1}$$

where:

$C_{storage} = T_{remain} \times D^{in} \times CostperStorage$ represents the cost of data storage where T_{remain} is the time that the data D^{in} is remaining in sc_i, D^{in} represents the stored data in sc_i and $CostperStorage$ is the cost of hosting time per second. For instance, the simple storage service Amazon S3 offers a range of storage classes designed for different use cases. There are three highly durable storage classes including Amazon S3 Standard for general-purpose storage of frequently accessed data, Amazon S3 Standard - Infrequent Access for long-lived, but less frequently accessed data, and Amazon Glacier for long-term archive [34]. The storage pricing of Amazon S3 varies by region. For instance, in the standard storage in the region of the US West (Northern California) for 50 TB/month, the price is \$0.026 per GB. However, in South America (Sao Paulo), the price is \$0.0405 per GB [34].

$C_{processing} = (T_{proc}+T_{wait}) \times CostperProcess$ indicates the cost of processing where T_{proc} is the time needed to process a dataset of a task, T_{wait} is the time waiting by a dataset to be processed and $CostperProcess$ is the cost of processing in million instructions per second ($MIPS$).

$C_{DataTransfer}$ is the cost of transferring data within region, across regions or over Internet. Consider sc_i a storage computing server which communicates with server sc_j, data transfer is charged for every gigabyte moved from sc_i to sc_j. Consider DT is the quantity of data to be transmitted between the two servers in gigabytes. The amount of data transferred is divided into slices, and each slice is defined by: (1) its size which is defined by an interval with max and min bounds, (2) its transfer cost depending on its size. For instance, Microsoft Azure uses several data size intervals according to data to be transferred, such as $[5TB - 10TB]$, $[50TB - 150TB]$, etc [58]. Hence, to be more closely to real environment, we propose a new way to compute data transfer cost. Indeed, we define the data transfer cost provided by a sc_i as a set S of n slices $S = \{s_i | i = 1, ..., n\}$. Indeed, each slice s_i is defined as a triplet $< s_i^{max}, s_i^{min}, p_i >$, with p_i is the price of the slice $\left[s_i^{max}, s_i^{min}\right]$. To note, for the latest slice, we have considered only s_n^{min}. For example, a data size over 500 TB, Microsoft Azure considered only the 500 TB [58]. Thus, the monthly data transfer cost is calculated as follows:

$$C_{DataTransfer}(sc_i) = \sum_{i=1}^{n-1} CostSlice(s_i) + (DT - s_n^{min})p_i \tag{2.2}$$

where: $CostSlice(s_i)$ represents the cost of data slice transfer from sc_i to any sc as indicated in the Eq. 2.3:

$$CostSlice(s_i(sc_i, sc_j)) = \begin{cases} (s_i^{max} - s_i^{min}) * p_i \text{ if } (DT_i^{'} > (s_i^{max} - s_i^{min})) \\ \text{otherwise }, DT_i^{'} * p_i \end{cases} \tag{2.3}$$

where $DT_i^{'}$ is the rest of data size for each used slice:

$$DT_i^{'} = DT_{i-1} - (s_{i-1}^{max} - s_{i-i}^{min}).$$

For instance, for the same data slice interval (5TB - 10TB), the prices of data transfer for the Central US is \$0.087 per GB, however, for the East Asia is evaluated as \$0.12 per GB for Microsoft Azure [58]. p_i depends also on the regions where the destination server is located. To note, a region is a geographic

location in which public Cloud providers' data centers reside [59]. For example, Amazon Web Services (AWS) operates regions in the United States, South America, etc [59]. Indeed, there's a cost of moving data across servers within the same region, and a different (generally greater) cost for data transfer across servers outside that region. Hence, according to the region, the cost is computed. First, if sc_i and sc_j are within the same region. Then, the traffic will not be charged and p_i is free. Second, if sc_i and sc_j are not in the same region, then costs apply and all outbound traffic is charged. This cost is denoted as $PriceAcrossRegion$. Finally, if the two servers communicate over Internet, so all internet traffic from sc_i to sc_j is bound to costs denoted as $PriceOverInternet$, which are the most greater. For instance, data going out of Azure data centers beloging to the interval (5GB - 10TB) to France Central are charged to \$0.084 per GB. Contrarily, for the same destination region and for the interval (50GB- 150Tb), the cost is evaluated as \$0.07 per GB [58]. (iii)$Scap : SC \rightarrow R^+$ is an available storage resource capacity function. $SS(sc_i)$ with $sc_i \in SC$ determine the maximum available storage capacity of sc_i in the CC environment. It may be measured in mega-bytes, giga-bytes or tera-bytes.

2.3. Data Center Network Architecture. In this scope, we have considered the three-tier Cisco data center network architecture [35] to avoid the problem faced with the two tier one when scaling up the number of servers, the network links in the core tier become over-subscribed. We have defined the link distance between sc_i and sc_j as *Physical Link Distance* ($PLD(sc_i, sc_j)$). Further, in order to determinate the communication cost, we assigned a *weight* for every (PLD). This link weight may be any practical measure such as link latency, number of hops or number of switches, etc. In our experiments, the PLD is defined as the number of switches on the routing path from source sc_i to destination sc_j denoted as $Nber_{\text{Switch}}$ and we have considered the fan-out of the access switches ($p0$) as well as the fan-out of the aggregation ones ($p1$) [36]. It can be expressed as follows [36]:

$$Nber_{\text{Switch}}(sc_i, sc_j) = \begin{cases} 0, \text{ if } i = j, \text{and} \\ 1, \text{ if } \lfloor \frac{i}{p_0} \rfloor = \lfloor \frac{j}{p_0} \rfloor, \text{and} \\ 3, \text{ if } \lfloor \frac{i}{p_0} \rfloor \neq \lfloor \frac{j}{p_0} \rfloor \wedge \lfloor \frac{i}{p_0 p_1} \rfloor = \lfloor \frac{j}{p_0 p_1} \rfloor, \\ 5, \text{ if } \lfloor \frac{i}{p_0 p_1} \rfloor \neq \lfloor \frac{j}{p_0 p_1} \rfloor, \end{cases} \tag{2.4}$$

$\lfloor x \rfloor$ is called floor function that gives the largest integer less than or equal to x [37]. $\wedge$ is a AND binary operator. In some cases, the communication between SC induces to increase the energy consumption in the communication elements (routers, switches, etc). Indeed, in a data center, the number of switches involved in the execution of a task is proportional to the position of SC to run the task [38]. Consequently, while examining the two active states and turned off of switch, the consumed energy by a network switch at times t, denoted as $Energy_{\text{Switch}}(t)$, is inspired from [38] and it is evaluated using the following formula:

$$Energy_{\text{Switch}}(t) = \alpha \times [P^{base}_{\text{Switch}}(t) + n_{activeport}(t) \times P^{activeport}_{\text{Switch}}(t)] \tag{2.5}$$

where $\alpha \in [0, 1]$; if $\alpha = 0$, it means that the switch is running, otherwise the switch is turned off. $P^{base}_{\text{Switch}}(t)$ represents the power consumed by the fixed parts of the Switch chassis and linecard and it is expressed as follows:

$$P^{base}_{\text{Switch}}(t) = P^{chassis}_{\text{Switch}}(t) + P^{linecard}_{\text{Switch}}(t) \tag{2.6}$$

$n_{activeport}$ is the number of active ports at time t. $P^{activeport}_{\text{Switch}}(t)$ represents the power consumed by an active port.

2.4. Problem Formulation. The large number of tasks in Intensive Workflow (IW) needs sometimes to process more than one dataset that may be stored in different SC. Moving these datasets increases the workflow execution time while consuming a high energy. As a result, the data movement between SC becomes a bottleneck that leads to limit the overall system performance [7] and increases the total cost of the system [39]. Figure 2.1 shows a sample scientific workflow instance that specifically describes our research questions. This scientific workflow consists of four tasks $\{t_1, t_2, t_3, t_4\}$, five input datasets $\{d_1, d_2, d_3, d_4, d_5\}$ initially existed in

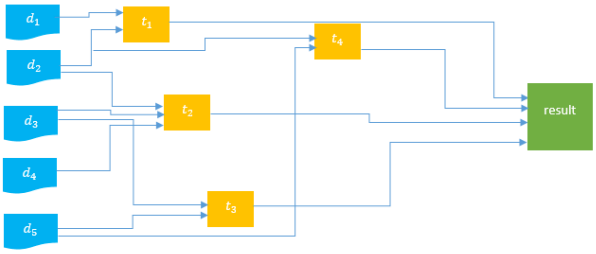


FIG. 2.1. *A Simple instance of a Workflow.*

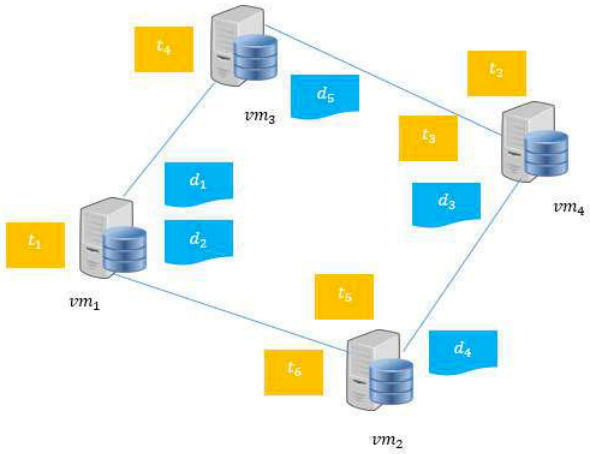


FIG. 2.2. *A virtual machine configuration in the Cloud with four virtual machines for workflow of Fig. 2.1.*

the system. In this figure, the data flows from dataset d_2 to t_1 and t_2 mean that d_2 will be used by both t_1 and t_2 and d_3 will be used by both t_2 and t_3. As shown in Fig.2.2, the task t_1 as well as the datasets d_1 and d_2 are assigned to the virtual machine vm_1. In same manner, the task t_4 and the dataset d_5 are assigned to vm_3. The tasks t_2 and t_3 and the dataset d_3 are placed in vm_4. Once the workflow tasks are executed, the task t_2 needs to transfer the datasets d_2 from vm_1 to vm_4 and d_4 from vm_1 to vm_4 to insure its execution. This data movement may increase the execution time and the energy consumption incurred during the communication between VMs.

2.5. Objective function. The data placement problem consists to providing a mapping of these datasets to SC that satisfies a set of objectives such as reducing the data transfer cost/time. Accordingly, given a workflow W, we formulate our data placement using mono-objective optimization problem formulation that aims to minimize the overall energy consumption and the cost of the workflow. The energy includes the communication and computing energy consumption of used devices. The cost of the workflow includes its data transfer cost, the data storage and the data processing cost. Hence, it is described by the Eq. 2.7:

$$
\begin{aligned}
&min((\textstyle\sum_{i=1}^{m}\sum_{j=1}^{s} x_{ij} * \omega_1 * Q_j) + \omega_2 * TDM(W)\\
&+\omega_3 * WorkflowEnergy)\\
&st.\\
&\forall i = 1..m, \forall j = 1..s, x_{ij} \in [0,1].\\
&\forall j = 1..s, \textstyle\sum_{i=1}^{m} size(i) \leqslant Cap(j)\\
&\forall i = 1..m, \textstyle\sum_{j=1}^{s} x_{ij} = 1\\
&\textstyle\sum_i \omega_i = 1\\
&\forall j = 1..s, \forall c = j..s-1, \forall i = 1..m, size(i) < BW(j,c)
\end{aligned}
\tag{2.7}
$$

s and m represent the number of SC and the number D^{in} respectively. x_{ij} is a boolean variable that indicates which dataset i is assigned to which server j. It is equal to 1 if the dataset i is assigned to the sc_j and to 0 otherwise. We assume that each dataset can be assigned to only one sc_j as indicated by Eq. 2.7. BW is the bandwidth demand or the traffic load between sc_j and sc_c. ω_1, ω_2 and ω_3 are the weight values that indicate the degree of importance of evaluated metrics of server which are described above, the total data movement and the total workflow energy consumption defined below and the costs parameters in the fitness function (Eq. 2.7) with their sum is equal to 1.

2.5.1. Total Data Movement for a given workflow W: $TDM(W)$. Consider the case where the dataset d_m^{in} is stored in sc_i. However, the the task t_x exists in sc_j. Actually, on assume that the d_m^{in} is required

to be transferred from sc_i to sc_j in order to satisfy the execution of the task t_x, hence, we define the Data Movement denoted as DM as follows:

$$DM(d_m^{\text{in}}, sc_i, sc_j) = \begin{cases} 0, \text{ if } sc_i \ and \ sc_j are \ in \ the \ same \ region \\ Size(d_m^{\text{in}}), \text{ if } sc_i \ and \ sc_j \ are \ in \ different \ regions \end{cases} \tag{2.8}$$

Therefore, the total data movement TDM for a given workflow W is defined as the total data transferred while executing the whole workflow tasks. It is calculated using the following formula:

$$TDM(W) = \sum_{t_x=1}^{n} \sum_{d_m^{\text{in}} \in DS(t_x)} DM(d_m^{\text{in}}, sc_i, sc_j) \tag{2.9}$$

n represents the number of tasks. When the servers sc_i and sc_j are in different regions, requested data must be transferred from sc_i to sc_j, so data traffic is generated accordingly.

2.5.2. Workflow Energy Consumption: $WorkflowEnergy$**.** Energy consumed during workflow execution includes both the energy consumed by network devices (CE) and computing and storage ($CompE$) devices, otherwise:

$$WorkflowEnergy = CE + CompE \tag{2.10}$$

Modeling the Workflow Communication Energy: CE**.** As the total amounts of workflow datasets can be very large, it may severely increase the consumed energy by the network devices (switches) and by the communication links. Thereby, communication energy depends on the total data being transferred, hence, the total data movement was included in the cost modeling of workflow communication consumption Energy. Recently, several works focused on controlling energy consumption of communication links in data center have been proposed aiming to reduce energy such as [11] [55] [62], etc.

In our paper, we propose a communication energy model which has to:
(i) Focus on a three-tiered data center network hierarchy described in Sect. 2.3,
(ii) Take into account the networking devices that consume a significant portion of overall energy consumption in data center in order to save energy,
(iii) Consider the consumed energy by the communication links, and
(iv) Operate at the granularity of the networking devices that may be either switch, router, etc.
For these reasons, our model is inspired from both works in [11] and [54] since the communication energy models proposed in these works meet our requirements. In our model, we assume that each computing storage server communicates with other servers through switches and a dedicated (contention free) reliable link that operates at the transmission rate of $R(i)(bits/s), i = 1, ..., s$ with s is the number of SC. Hence, our communication energy model is expressed as follows:

$$CE = \sum_{i=1}^{s-1} \sum_{j=i+1}^{s} cost(sc_i, sc_j) * DM(d_s^{\text{in}}, sc_i, sc_j) + \sum_{c=1}^{s} \varepsilon_{\text{net}}(c) \tag{2.11}$$

where $cost(sc_i, sc_j)$ is the function computing the communication cost between any two SC. It is defined as the product of the number of switches on the routing path from sc_i to sc_j as described in Eq. 2.4 and the consumed energy by each switch as indicated by Eq. 2.5. Formally, it is expressed as:

$$cost(sc_i, sc_j) = Nber_{Switch}(sc_i, sc_j) \times Energy_{\text{Switch}}(t) \tag{2.12}$$

$\varepsilon_{\text{net}}(j)$ is the consumed energy by the one-way transmission plus switching operation is described in the following formula:

$$\varepsilon_{\text{net}}(j) = P_{\text{net}}(j) \times \frac{size(d_j^{\text{in}})}{R_j} \tag{2.13}$$

$D(j) = \frac{size(d_j^{\text{in}})}{R_j}$ is the delay corresponding to the one-way transmission with $size(d_j^{\text{in}})$ is the dataset size to be moved to other SC and $R(j)$ is the transmission rate. P_{net} is the power consumed by the one-way transmission plus switching operation which depends on the corresponding transmission rate $R(j)$, the bandwidth $BW_j(Hz)$, the noise spectral power density $N_0(j)(BW/Hz)$, (non negative) gain g_j of the j^{th} link and the power consumed by the j^{th} end-to-end connection in the idle mode P_{idle} [29].

$$P_{\text{net}}(j) = \xi_j(2^{\frac{R(j)}{BW_j}} - 1) + P_{idle}(j) \tag{2.14}$$

with $\xi_j = \dfrac{N_0(j) \times BW_j}{g_j}, j = 1, ..., s$

Modeling the Workflow Computation Energy: $CompE$**.** Several approaches have been proposed to handle reducing computing energy consumption [11] [54] [56], etc. For the computation energy model, we have applied the high level approach and it is inspired from [54] since their model is general and it avoids relying on only processor element which increase the portability of the model and the simulation speed [57]. Hence, our power consumption of a computing server or CPU is linearly related to the server's utilization rate α and the associated power consumption.

$$CompE = (1 - \alpha) \cdot P_{CPU_{idle}} + \alpha \cdot P_{CPU_{Full}} \tag{2.15}$$

where $\alpha \in [0, 1]$ represents the utilization rate of CPU, being $\alpha = 0$, the machine is in an idle state and the associated power consumption is $P_{CPU_{idle}}$, whereas $\alpha = 1$ the machine is in a full state and the associated power consumption is $P_{CPU_{full}}$. For the idle state, the power consumption $P_{CPU_{idle}}$ is expressed as:

$$P_{CPU_{idle}} = A \cdot C \cdot f_{idle} \cdot v_{idle}^2 \tag{2.16}$$

where V_{idle} and f_{idle} denote respectively the voltage V and frequency f when the CPU is in idle state. The power consumed by CPU in the full state $P_{CPU_{full}}$ corresponds to:

$$P_{CPU_{full}} = A \cdot C \cdot f_{full} \cdot v_{full}^2 \tag{2.17}$$

where V_{full} and f_{full} are the voltage V and frequency f when the CPU is in full state.

3. Proposed solution: E-DPSIW-FCA. Our data placement strategy targets at finding a minimal number of SC where the maximum of datasets and tasks are grouped based on their dependencies. It is based essentially on the FCA approach. Note that reader can review the paper [15] to fully understand the mathematical foundations of FCA approach. In the following, we will justify our choice for this approach.

3.1. Choice of the FCA approach. The choice of FCA approach is motivated by the following reasons: (i)Apart from the mathematical foundation of the FCA approach, the notion of the concept represents faithfully the notion of tasks grouped based on their common attributes. These attributes represent the common datasets that tasks need for their execution, (ii)Because FCA results could be manipulated by some operators to navigate in Galois-lattice structure, we need it to consider relationships among concepts, (iii)Its effectiveness to deal with the problem faced during the placement of data when the group data size exceeds the storage capacity of SC, that is considered as a compromise for prior data placement strategies. However, using the hierarchy between the lattice concepts, this problem could be resolved by navigating among the different concepts. Furthermore, to understand our solution, we need first to mention the necessary used definitions.

3.2. Definitions.

Extension, Intention and Size of a Concept c_i**:** This definition was given in [15]. Consider C is a set of p Lattice concepts. Each concept c_i is defined by the couple: $c_i =< Ext, Int >$; with Ext is called the extension of the concept which is the tasks group. Int is called the intent of the same concept which represents the input data. $c_i.Int$ refers to the set of data existed in the intent of the concept c_i. The size of the concept c_i is defined as:

$$Size(c_i) = \sum_{d_i^{\text{in}} \in |c_i.Int|} Size(d_i^{\text{in}}), d_i^{\text{in}} \in D^{\text{in}} \tag{3.1}$$

Sparse Formal Context: The set of Tasks T and the set of original datasets D^{in} are used in order to generate the Formal Context. Indeed, a formal context that contains more number of Zero elements than non-Zero elements is known as *Sparse Formal Context*. More specifically, to check whether a formal context which has O as the number of objects (or entities), A as the number of attributes (or properties) is *sparse*, we need to verify the total number of zero. If this count is more than $(O * A)/2$, we consider this context as *sparse*, otherwise, it is considered as *dense* context.

Level of concept ($Level(c_i)$): In order to improve the solution execution time in [15], we propose to use the level of a concept in a Galois-lattice. It's is defined as the cardinality of its intention in our case. The level of a concept c_i is determined as:

$$Level(c_i) = |c_i.Int| \tag{3.2}$$

It's necessary to note that while navigating in the lattice from the top going bottom, the level of concepts increases. The high level in a Galois-lattice is defined as the maximum cardinality of the all concepts intention. In fact, a high level in a lattice of concepts is described as:

$$High_{level} = max_{j=1}^{p}(Level(c_j)) \tag{3.3}$$

p is the concepts number in the Lattice. Note that the concepts located in the lattice middle have the highest level.

Weight of a Concept ($P(c_i)$): We improved the weight definition in [15] by considering in this scope the granularity of the data center resources. Specifically, we have replaced the data center with the computing storage server in the data placement. Besides, the weight $P(c_i)$ of a concept is denoted as follows:

$$P(c_i) = \frac{|c_i.Ext|}{|T|} \times \frac{|c_i.Int|}{|D^{\text{in}}|} \times \frac{MaxCap(sc_i) - Size(c_i)}{|MaxCap(sci) - Size(c_i)|} \tag{3.4}$$

The measure of the concept weight allows us to have the concepts that contain a maximum number of tasks and datasets. The third factor is used to verify if the concept exceeds c_i the maximum storage capacity of sc_i.

Minimum Data Coverage (MDC): Let $C = \{c_1, c_2..., c_p\}$ the set of p concepts. C is MDC if it covers all datasets. Formally, we define MDC of C by:

$$MDC(C) = \begin{cases} 1, \text{ if } \bigcup_{1 \leq i \leq p} c_i.Int = D^{\text{in}} \text{ and } \bigcap_{1 \leq i \leq p} c_i.Int = \oslash \\ 0, \text{otherwise} \end{cases} \tag{3.5}$$

Candidacy of a Concept c_i ($Candidate(c_i)$): A concept c_i is called candidate concept, if it has a maximum weight and it covers the whole datasets. We define the candidacy of c_i as follows:

$$Candidacy(c_i) = \begin{cases} 1, \text{ if } P(c_i) \text{ is maximum and } MDC(c_i) = 1 \\ 0, \text{ otherwise} \end{cases} \tag{3.6}$$

Feasibility of a Concept c_i (Feasibility (c_i)): A concept is feasible if the total size of its datasets is small enough to be assigned to one storage computing server to be able to assign them. It is defined as follows:

$$Feasibility(c_i) = \begin{cases} 1, \text{ if } \exists sc_i \text{ where } \sum_{d_i^{\text{in}} \in cc_i.Int} \\ Size(d_i^{\text{in}}) \leqslant Cap(sc_i) \\ 0, \text{ otherwise} \end{cases} \tag{3.7}$$

Score of Dataset d_i^{in} (Score (d_i^{in})): For each dataset $d_i^{\text{in}} \in D^{\text{in}}$, let $SC_{d_i^{\text{in}}}$ denotes the set of SC that contained d_i^{in}.Thus, the score of dataset d_i^{in} is defined as:

$$Score(d_i^{\text{in}}) = |\{t_i \in T \text{ such as } R_{t_i} \cap SC_{d_i^{\text{in}}} = \{d_i^{\text{in}}\}\}| \tag{3.8}$$

R_{t_i} is a partial order relation which gives all the datasets used by a task t_i. Different from the solution in [15], the datacenter in the feasibility formula (3.7) and the score definition (3.8) was replaced by the storage computing server.

Tasks dependency $dependency^T(t_i, t_j)$: $dependency^T(t_i, t_j)$ represents the dependency between the tasks t_i and t_j. To note, two tasks are dependent if they use the same datasets to satisfy their execution. Consequently, the number of dependency between the tasks t_i and t_j is the number of data that are used by both t_i and t_j which is defined as follows:

$$dependency^T(t_i, t_j) = Count(D^{\text{in}}(t_i) \cap D^{\text{in}}(t_j)) \tag{3.9}$$

where: $D^{\text{in}}(t_i) \subseteq D^{\text{in}}$, $D^{\text{in}}(t_j) \subseteq D^{\text{in}}$ are the datasets that the tasks t_i and t_j respectively require to execution. $Count$ returns the number of data used by both the tasks t_i and t_j.

Quantity of transferred data: (QtDT(d^{in}_{com}, c_i, c_j)) Consider two concepts c_i and c_j, and d^{in}_{com} a common original dataset used by the two concepts: $d^{\text{in}}_{com} = \{c_i.Int\} \cap \{c_j.Int\}$. We have to verify the condition of the minimum data coverage $\bigcap_{1 \leq i \leq p} c_i.Int = \oslash$ which represents the set of elements which are in c_i or c_j, or in both c_i and c_j. Then, we have to decide where to place the common dataset between the two concepts c_i and c_j. Indeed, our choice is based essentially on finding the best placement which requires a minimum of: (i)energy consumed during communication as described in Eq. 2.11, (ii)total data movement (Eq. 2.9) hopping through network devices that will eventually reduces the network overhead. The quantity of transmitted data between two concepts is described in the Algorithm 1 where sc_i, sc_j are the concepts' computing storage servers c_i and c_j. $Count(t_i)$ and $Count(t_j)$ are the number of tasks presents in the concepts c_i and c_j respectively that require the common dataset d^{in}_{com}. $DM(d^{\text{in}}_{com}, sc_i, sc_j)$ and CE are described above in (Eq. 2.8) and (Eq. 2.11).

Algorithm 1 Quantity of transferred Data function

function QtDT($d^{\text{in}}_{com}, c_i, c_j, sc_i, sc_j$) ▷ $d^{\text{in}}_{\text{com}}$ is placed in c_i
  if $d^{\text{in}}_{com} \in \{c_i.Int\}$ **then**
    $QtDT(d^{\text{in}}_{com}, c_i, c_j, sc_i, sc_j) = \sum_{d^{\text{in}}_j \in c_j.Int} Size(d^{\text{in}}_j) - Size(d^{\text{in}}_{com})) \times (Count(t_j))t_j \in c_j.Ext(t_j) + DM(d^{\text{in}}_{com}, sc_i, sc_j) + CE$
  end if
end function

Overall, we aim to place the independent data in a minimum number of SC, subsequently, in the lowest level of communication in order to reduce the data movement between SC and minimizing the workflow communication energy. Indeed, finding the appropriate storage computing server to place d^{in}_{com} among sc_i and sc_j described by the Algorithm 2.

Algorithm 2 Appropriate Placement's Algorithm

function Placement($QtDT(d^{\text{in}}_{com}, c_i, c_j, sc_i, sc_j)$, $QtDT(d^{\text{in}}_{com}, c_j, c_i, sc_i, sc_j)$)
  Output: Var *AppropriatePlacement*
  if QtDT($d^{\text{in}}_{com}, c_i, c_j, sc_i, sc_j$) < QtDT($d^{\text{in}}_{com}, c_j, c_i, sc_i, sc_j$) **then**
    $AppropriatePlacement = sc_i$
  else
    $AppropriatePlacement = sc_j$
  end if
end function

3.3. Proposed Data Placement Solution. In our previous work, a data placement strategy to dataset-datacenter mappings was proposed. It was just a start towards using FCA for the data placement problem. Based on this work, we have further investigated the characteristics of FCA approach to improve our previous work steps by: (i)classify input workflows and reduce formal context in order to save time when building the

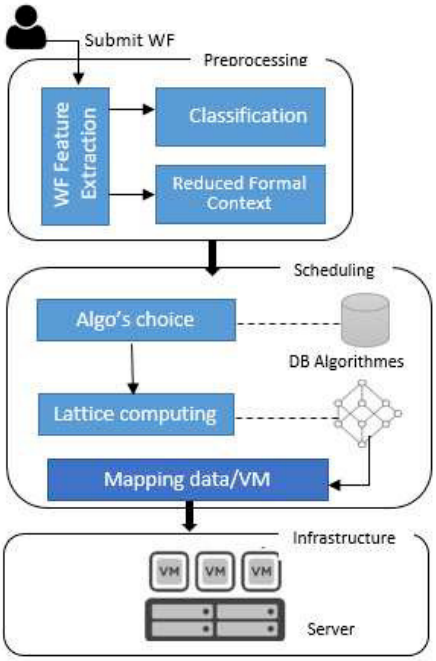


FIG. 3.1. *Overview of our proposed solution E-DPSIW-FCA.*

lattice, and (ii)exploit the lattice concepts level notion to define an heuristic in order to prevent analyzing all the concepts during the candidate concepts selection. Fig. 3.1 presents the architecture of E-DPSIW-FCA:

3.3.1. Workflow Features Extraction. We have resorted to use an XML parser to parse the workflow features, which is stored in XML format. Indeed, the XML parser extracted all workflows tasks, a set of input datasets required for their execution and their sizes and also the dependencies between these tasks.

3.3.2. Pre-processing. We improved our previous work by adding this new step, which consists of two phases:

Workflows Classification: In this stage, we classify the input workflows in order to identify the most convenient techniques or algorithms to be used for the generation of Galois-lattice. Thus, we have to consider the characteristics of the input workflows for the choice of these algorithms. Indeed, these characteristics include the following metrics: the number of datasets, and the dependency between the tasks of the workflow [63]. Based on data dependencies between the tasks, we defined a set of metrics to initially classify the input workflows as workflows with high dependency and workflows with low dependency. Noteworthy that this classification allow us identifying the dense formal context and this is in the case of high dependency workflows and the sparse formal context in the case of ow dependency workflows. Further, based on the types of these formal context, the lattice construction algorithms are recommended. For example, the Godin algorithm [43] has a good performance with a sparse formal context. Note that we have used a clustering algorithm to be able to automatically classify the input workflows.

High Dependency Workflows: This category of workflows includes multiple dependencies between their tasks, since these tasks require for their execution the same datasets. These dependencies are reflected by intensive data transfer between tasks. This type of workflows generates a significant execution time and high-energy consumption.

Low Dependency Workflows: These workflows contain essentially a few number of dependent tasks (See Def. 3.9). Thus, the amount of data communication between these tasks is quite small. This type of workflows maximizes the parallelism of tasks execution which allows saving both the total execution time and energy consumption. For instance, we consider the Eq. 3.10 to identify whether the input workflows with high

dependency or with low dependency.

$$\frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} dependency^{T}(t_i,t_j)}{n} \geq Limit_{dependency} \tag{3.10}$$

where n is the overall number of tasks. $dependency^{T}(t_i,t_j)$ is defined above (def. 3.9). $Limit_{dependency}$ is the dependency threshold between tasks. In fact, beyond this threshold, one can consider the workflow with a high dependence between tasks and below, the workflow is considered with a low dependence. Noteworthy that is the formal context represents the data dependency relationships of a input workflow.

The performance of Lattice generation algorithms depends essentially on the formal context's density. Thus, a K-means based algorithm [62] that allows clustering the input workflows based on their feature (datasets, tasks, tasks dependencies) is used. As described in [63], this algorithm allows us to identify the most convenient algorithms to apply for the lattice concepts generation. K-means aims to partition an input set of N points into K clusters by finding a partition such that the squared error between the empirical mean of a cluster and the points in the cluster is minimized. The squared error metric and more details about the K-means algorithm could be found in [62]. The main steps of K-means algorithm are as follows:

1. Select an initial partition with K clusters; repeat steps 2 and 3 until cluster membership stabilizes,
2. Generate a new partition by assigning each pattern to its closest cluster center, and
3. Compute new cluster centers.

This algorithm requires three user-specified parameters: number of clusters K, cluster initialization, and distance metric. The number of clusters in this work is limited to two defining the type of the workflow dependency (high, low). The definition of initial clusters is accomplished using the Eq. 3.10 determining a first classification of the workflows. Then, we use the graph distance measure defined in [62] to evaluate the similarity degree between the DAG graphs of the input workflows.

Reducing Formal Context. From the set of Tasks T and the set of original datasets D^{in} extracted in step 3.3.1, we generate the Formal Context $FC = (T, D^{\text{in}}, I)$. I is the binary relationship between T and D^{in}, it is set to 1 if task t_i needs the dataset d_i^{in} for its execution, 0 otherwise. Further, the Formal Context indicates the dependency between tasks. Tow tasks t_i and t_j are dependent if only if they share one or more datasets. This step is new compared to our previous solution. Since the concept of Formal context is the central of FCA approach and its size has a significant influence on the structure of concept lattice as well as time and space complexity [60] when building the lattice, hence, it is important to reduce this formal context. In general, many techniques were proposed for this purpose. Thus, we have applied a simple way of reduction, which was suggested by [42]. Its principle consists of the junction of lines and/or columns. Indeed, if for two objects g and h are equal ($g = h$) then g and h can be replaced by one single object; dually, if for two attributes m and n are equal ($m = n$), then m and n can be replaced by one single attribute [42].

3.3.3. Choice of the lattice Construction Algorithm. According to [61], the lattice construction algorithms are recommended in terms of density/sparseness of underlying formal contexts as follows: (i)When the formal context is **small** and **sparse** as denoted in 3.2 **Godin** [43] algorithm is a good choice in this case, (ii)However, when contexts become **denser**, the algorithms such as **Norris** [44], **NextClosure** [45] and **Close by One** [46] should be applied, and (iii)Though, in case of *average density* context, **Bordat** [47] algorithm performs well.

3.3.4. Building of Galois-lattice. Our claim is to group the maximum of datasets and tasks together based on their dependencies. This group is called Formal Concept referred to FCA. Moreover, in order to represent all the formal concepts and their relationships, we build the Galois-lattice from the reduced context $FC = (T, D^{\text{in}}, I)$ and depending on the generation algorithm chosen in the above step. Then, we proceed to the reduction of the initial generated Galois-lattice in order to find the minimum number of concepts that have grouped a maximum datasets and tasks and eliminate the unnecessary and redundant concepts without loss of the information. In our solution, we eliminate all concepts having a single intention (or null) since the tasks belonging to these concepts require only one dataset to be processed. The recovery of this dataset is possible from other lattice concepts.

3.3.5. Choice of Candidate Concepts based on Concept Level. It refers to extract from the Galois-lattice, all the candidate concepts that, together, have a maximum weight and can cover the entire set of attributes (D^{in}). Nevertheless, we have proposed in this paper a new heuristic, which exploit the notion of level defined in 3.2 in order to avoid examining all of them. Actually, we will firstly organize all the concepts by level in the Lattice and then choose the concepts existed in the middle since they have the highest weight and cover all the data. After selecting concepts and before affecting them into appropriates SC, we have to verify if two candidate concepts, which are kept, have a common dataset. If it is the case, we will choose the placement of the common dataset in one storage computing server of the candidate concepts by comparing the quantity of transferred datasets in the two cases according to the algorithms1 and 2.

3.3.6. Mapping Datasets to Storage Computing Servers. In this paper we will take into account the granularity of the utilized resources in the data center and its different communication levels during the placement of IW data. Its basic is to assign the datasets of each concept candidate cc_i (See Eq. 3.6) and feasible (See Eq. 3.7) to the appropriate sc_i to prove that the difference between its storage capacity and the size of the concept cc_i is the minimum among every $sc_i \in SC$. If any concept among the candidate concepts isn't more feasible with what is left as free SC, we will proceed to assigning its sub-concepts denoted as $SubC(cc_i) = \{scp_1, scp_2, ..., scp_s\}$. We treat this sub-concepts in the following manner: we select the small size dataset as the first sub-concept to be placed later in the appropriate sc_i in order to avoid the transfer of larger datasets. Then, we put the rest of datasets in a second sub-concept. In this data assigning step, we proceed to the elimination of already assigned data from others sub-concepts in order to prevent their redundancy. The stop condition is that all the datasets of cc_i are placed ($cc_i.Int = \oslash$) and $\forall cc_i \in SubC(cc_i)$, we have $scp_i.Int = \oslash$. Finally, the result of this step is the list m of SC where the datasets are allocated.

3.3.7. Data Replication. The data replication technique is considered in our approach as a solution to ensure the minimization of the average cost of all tasks. This technique was detailed in our previous work [15]. Briefly, it is based on two steps which are: i) Identification of data to replicate based on the score of each dataset $Score(d_i^{\text{in}})$ (Eq. 3.8), and ii) Replication of important datasets.

3.4. E-DPSIW-FCA algorithm. The E-DPSIW-FCA algorithm is outlined in Algorithm 4. The algorithm 4 describes the proposed strategy. In the first step, we start by classify the input workflows by applying the K-means algorithm (KmeansCluster). Then, we generate the formal context $FContext$ from the workflow tasks and datasets using *generateFormalCxt* function. We apply the algorithm suggested in [42] for the reduction of the generated Formal $FContext$ using *reduceFCxtbyGanter(FContext)* to have our $ReducedFCxt$. We generate the Lattice Galois from $ReducedFCxt$ by applying the appropriate classification algorithm as described above by the use of *generateLattice* function. This generated lattice, in turn, will be simplified using *simplifyLattice* function. We retain then the candidate concepts from this simplified lattice by the use of *selectCpCand* function. In the next step, for each candidate concept, we verify its feasibility using *feasible* function. If the concept is feasible, we first apply the function *findAppropriateServer* to find the most appropriate storage computing server where we store this datasets concept using the function *affecte*. Then, we add this server to the best servers list. If the concept isn't feasible, we find its sub-concepts using *findSubCpts* function and then we update the list of candidate concepts with the function *update(CandidateConcepts, SubConcepts* to perfom the same treatment again. In the last step, for each storage computing server $sc_i \in Placement$, we verify if its datasets need to be replicated using the function *score*. If it is the case, the replication is done by the use of *replicate* function. $WF.D^{\text{in}}$ stands for the data D^{in} of the input workflows WF.

4. Experiment Simulation. In our experiments, we run a set of scientific workflows as the sample tests from [48] notably earthquake science (*CyberShake*) [49], astronomy (*Montage*) [50] and biological genetics (*Epigenomics*) [51]. They were chosen because they represent a wide range of application domains and they have distinct structures and differ greatly in the number and size of datasets. For example, Montage is I/O intensive, CyberShake is memory intensive, and Epigenomics is CPU intensive. The Montage project is an astronomy application that delivers science-grade mosaics of the sky [50]. The CyberShake is used to calculate Probabilistic Seismic Hazard curves for several geographic sites in the Southern California area [52]. Epigenomics maps short DNA segments collected with high-throughput gene sequencing machines to a previously structured reference genome [51]. We have used the execution traces of the tested workflows in [48], which are stored in

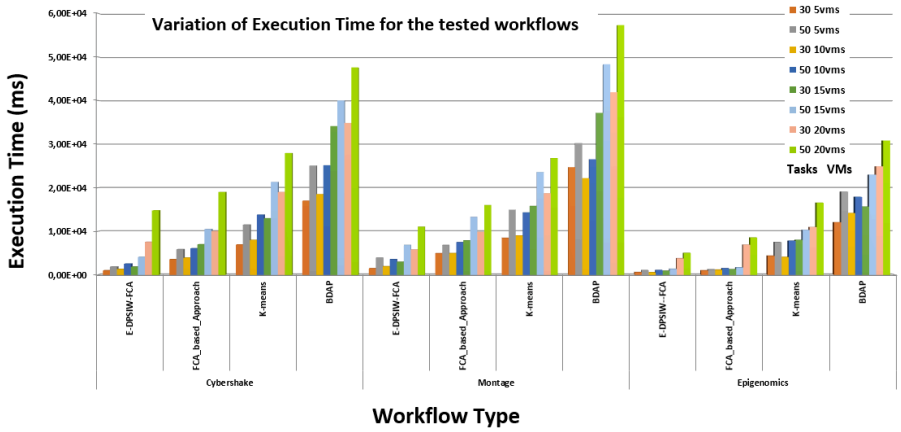
Algorithm 4 E-DPSIW-FCA Algorithm

Input: WF ▷ The input Workflow
$setofServers = sc_1, sc_2, ..., sc_n$ ▷ List of SC
$CandidateConcepts = \oslash$ ▷ List of candidate concepts
$SubConcepts = \oslash$ ▷ List of sub-concepts
$FContext = \oslash$ ▷ Formal context to generate
$ReducedFCxt = \oslash$ ▷ Reduced Formal context
$lattice = \oslash$ ▷ Galois-lattice to build
$simplifiedLattice = \oslash$ ▷ Galois-lattice simplified
Output: Var $Placement$ ▷ List of couple $< sc_i, d_j >$

int $capacity = 0$;
$intworkflow - type = KmeansCluster(WF)$;
$FContext = generateFormalCxt(intworkflow - type.T, intworkflow - type.D^{\text{in}})$;
$ReducedFCxt = reduceFCxtbyGanter(FContext)$;
$lattice = generateLattice(ReducedFCxt)$;
$simplifiedLattice = simplifyLattice(Lattice)$;
$CandidateConcepts = selectCpCand(simplifiedLattice)$;
for $candidate \in CandidateConcepts$ **do**
  if $feasible(candidate)$ **then**
    $sc_j = findAppropriateServer(sc_i, setofServers)$;
    $capacity = Cap(sc_j)$
    $capacity = (min(Cap(sc_i),\ size(candidate))$ ▷ $\forall i = 1..n.$
    $affecte(candidate, sc_j)$
    $add(sc_j, Placement)$
    $sc_j.Cap = sc_j.Cap - Size(candidate)$
  else
    $SubConcepts = findSubCpts(candidate)$
    $update(CandidateConcepts, SubConcepts)$;
    for $sc_i \in Placement$ **do**
      for $d_i \in sc_i.WF.D^{\text{in}}$ **do**
        if $score(d_i^{\text{in}})$ **then**
          $replicate(d_i^{\text{in}})$
        end if
      end for
    end for
  end if
end for

XML-formatted files while providing information about these workflows, such as datasets size and dependencies, tasks flow, etc.

4.1. Simulation Setting. Our simulation environment was developed using CloudSimPlus toolkit an extension of CloudSim [53], which provides a platform of Cloud Computing infrastructures. The same workflows and CC environments are simulated on other contrast experiments described in the Sect. 5 which are the BDAP in [14] and the K-means strategy [4]. According to [14], the parameters of BDAP approach based GA are set as follows: the population size is 20, the maximum number of iterations is 20, the crossover probability is 0.8, the mutation probability is 0.5. In our experiments, we assume that the weight values that indicate the degree of importance of the quality metrics are equals. Our solution E-DPSIW-FCA, K-means and BDAP were implemented and evaluated for the three test workflows according to the following criteria: 1)Execution time of workflow tasks, 2)Workflow Communication energy cost that is defined in Sect. 2.11, 3)Total data movement defined in Sect.2.9, and 4)Workflow Computing energy which is already defined in Sect. 2.15. We did our



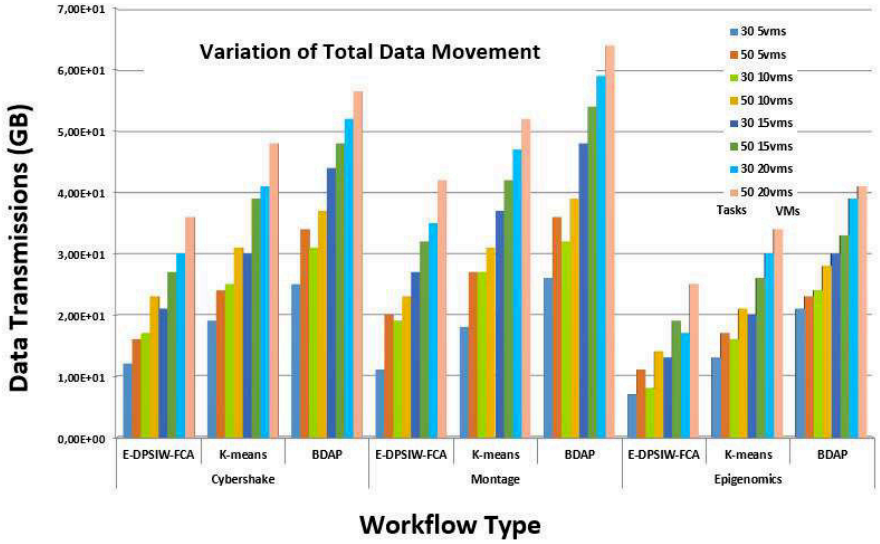
(a) 30 and 50 tasks.

(b) 100, 200 and 300 tasks.

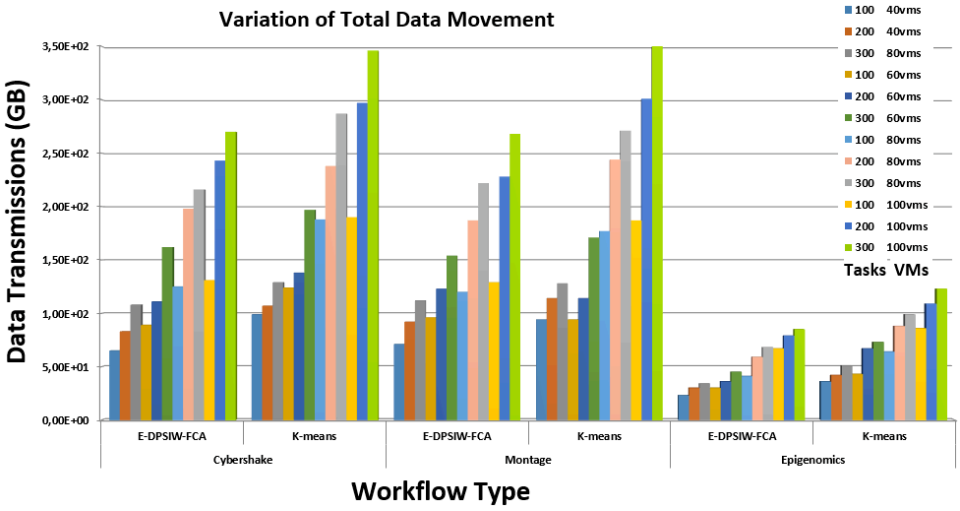
FIG. 4.1. *Total execution time by varying the number of Tasks.*

experiments for two different scenarios: firstly with varying the number of tasks and secondly with varying the number of VMs aiming to observe the effect of these variation on our performance indicators.

4.1.1. Scenario 1: Varying the number of Tasks. In this scenario, we have considered firstly four size number of VMs: 5, 10, 15 and 20 for the E-DPSIW-FCA, BDAP and K-means approaches. In this test, we select for each tested workflow a number of tasks in the range [30, 50]. Then, we test for E-DPSDIW-FCA and K-means, since the processing time of the genetic algorithm (BDAP) approach is higher and it trends to take longer to converge upon a solution. The number of VMs is in the range [40, 60, 80, 100, 120, 140, 160, 180, 200] with number of tasks set at 100, 200, 300 tasks. Note that each task can be executed on any virtual machine. With reference to Fig. 4.2, it can be seen that our solution has managed to reduce the total data movement between VMs compared to BDAP and K-means algorithms. This reduction is more valuable for the memory intensive workflows (Cybershake) and less remarkable in the case of CPU intensive workflows (Epigenomics) since its amount of data communication is quite small. Figure 4.3 confirms the ability of our strategy to reduce heavily the energy consumed during the communication between SC. Comparing the experiment results of K-means, BDAP and our strategy, in Fig. 4.4, in terms of Computing Energy Consumption, it would be found that the performance of our approach is the better.



(a) 30 and 50 tasks.

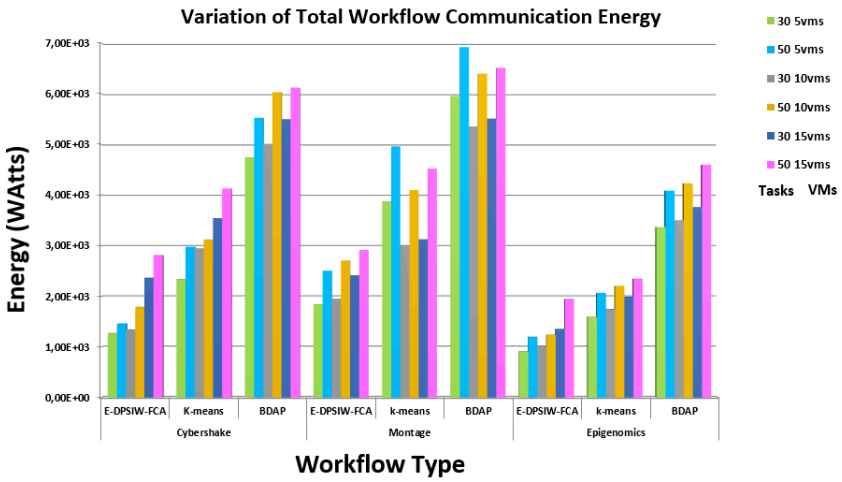


(b) 100, 200 and 300 tasks.

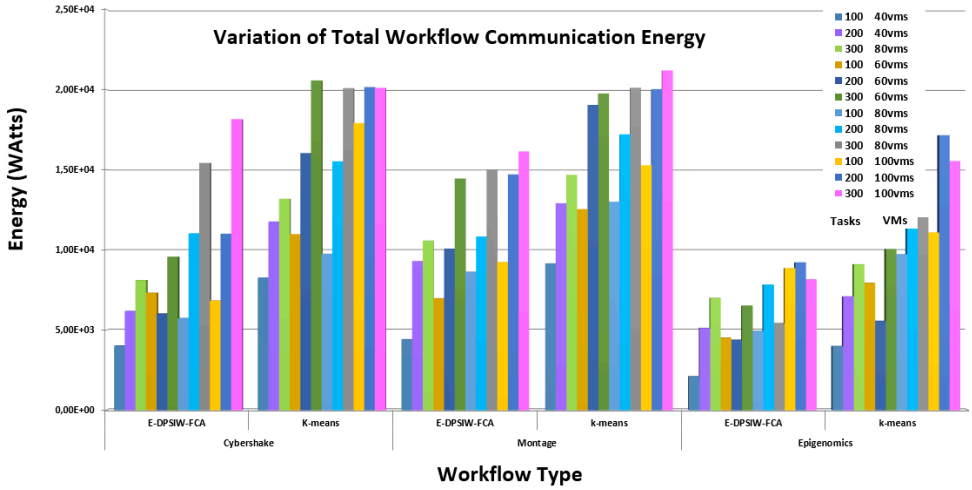
FIG. 4.2. *Total Data movement by varying the number of Tasks.*

4.1.2. Scenario 2: Varying the number of VMs. In this scenario, we did our experiments for the E-DPSIW-FCA and K-means with fixing the number of tasks to 300 and the number of VMs was set to [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000]. It is worth noting that BDAP trends to take longer time to converge upon a solution and it blocks at 100 tasks, so it was excluded. Figure. 4.5 reveals well that as the number of VMs increases, FCA-based-Approach, K-means, and BDAP strategies' respective execution time prove to record a noticeable increase as compared to our approach. Further, it is clearly seen that our strategy reduces heavily the execution time for Epigenomics better than Cybershake, since Epigenomics maximizes the parallelism of tasks while saving time. Additionally, our approach performs well the execution time compared to our previous work FCA-based-Approach, especially in the case of Cybershake rather than Epigenomics.

Figure 4.6 exhibits the total data movement' tendencies recorded. In fact, our approach reduces the amount of data movement compared to other strategies. This result has its justification, since our approach avoids the movement of larger data and it provides multiple data placement choices, thereby, decreasing the amount of



(a) 30 and 50 tasks.



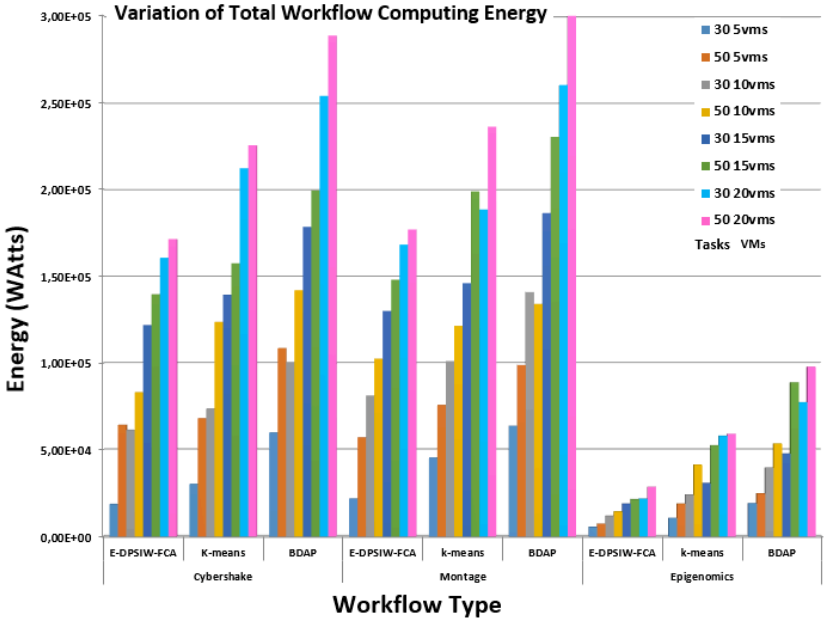
(b) 100, 200 and 300 tasks.

FIG. 4.3. *Total Workflow Communication energy by varying the number of Tasks.*

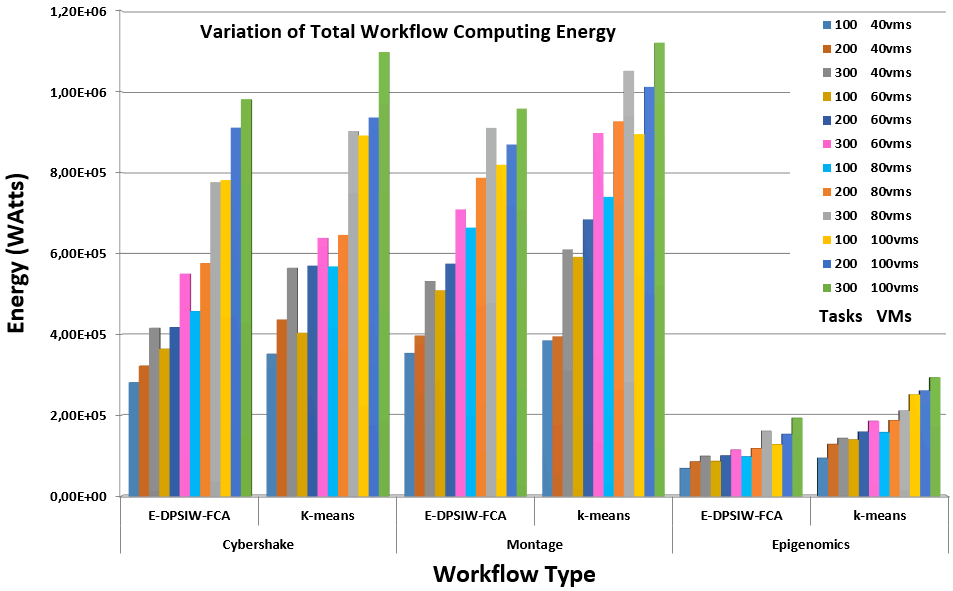
data transfer consequently. However, K-means considers data dependencies too much, which leads to larger data movement. Figure 4.7 indicates the additional energy consumption values incurred from the communication between VMs. As data movement between VMs leads to a communication energy consumption, based on this, it can be seen that our approach reduces the communication energy noticeably to BDAP and K-means. Worth citing that data movement reduction and the communication energy consumption decrease is more valuable for the memory intensive workflows (Cybershake and Montage) and less remarkable in the case of CPU intensive workflows(Epigenomics), since its amount of data communication is quite small. Similarly, as shown in Fig. 4.8, our approach reduces the computing consumption compared to other strategies.

5. Related Works. Being a critical challenging issue, data placement of IW has always attracted the attention of researchers. Several data placement strategies have been proposed in this issue and tend to be categorized under data dependency and graph-based methods.

Data dependency approaches have attempted to solve the data placement problem via application of the heuristic algorithms such as the Genetic Algorithm (GA), the Practical Swarm Optimization technique (PSO), the Ant Colony Optimization (ACO) and the hierarchical partition algorithms (based on Bond Energy Algorithm (BEA)) to group data based on their dependencies in a bid to reduce the total data movements



(a) 30 and 50 tasks.



(b) 100, 200 and 300 tasks.

FIG. 4.4. *Total Workflow Computing energy by varying the number of Tasks.*

throughout the workflow execution processes, such as [14] [17] [20], etc.

Graph-based approaches used, the data placement problem by modeling as a hypergraph or graph such as in [21], [22], [23] and [31]. Furthermore, we have identified a set of criteria in order to perform a comparison study (Table 5.1) of the data placement strategies above-mentioned which are:

(i) *objective* - which optimization criteria are exploited,

(ii) *technique used*- which concept is used to indicate the type of algorithm or method employed,

(iii) *modelization*- how authors model the data placement problem (graph, matrix),

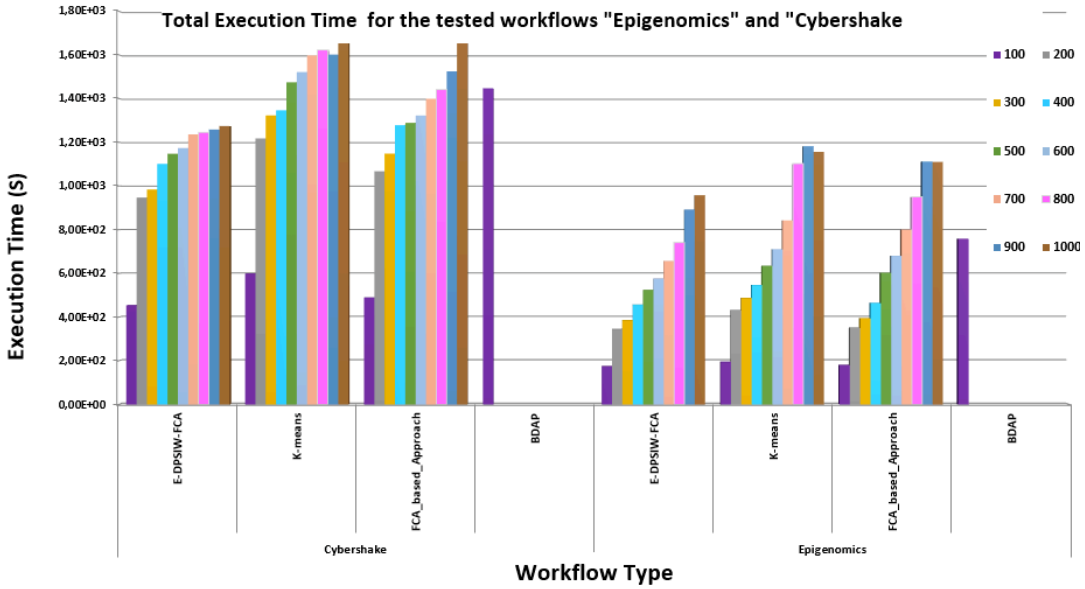


FIG. 4.5. *Total Execution Time by varying the number of VMs.*

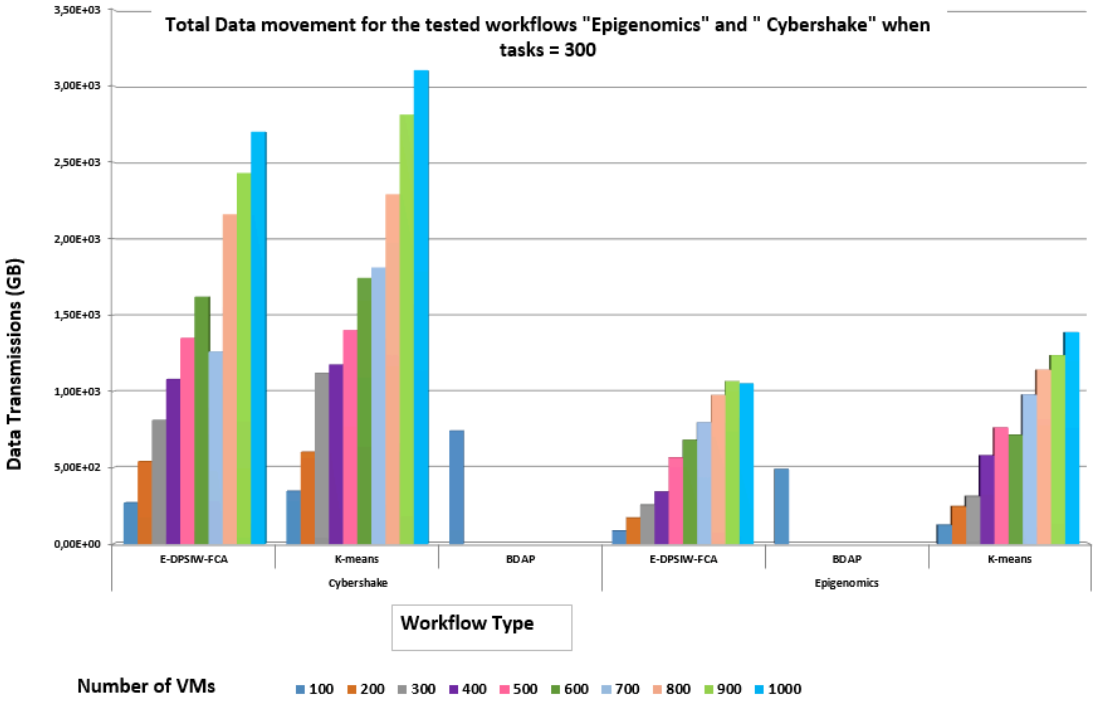


FIG. 4.6. *Total Data movement by varying the number of VMs.*

(iv) *type of datasets*- datatsets can be original (*Inp*), intermediate data *Int*), produced by running workflow, or fixed data *Fix*, stored in specific data centers according to their size or their management needs, flexible data *Flex*, that the system can flexibly decide where to store them,

(v) *data center infrastructure*- what is the granularity of the resources considered (virtual machines, servers, etc.),

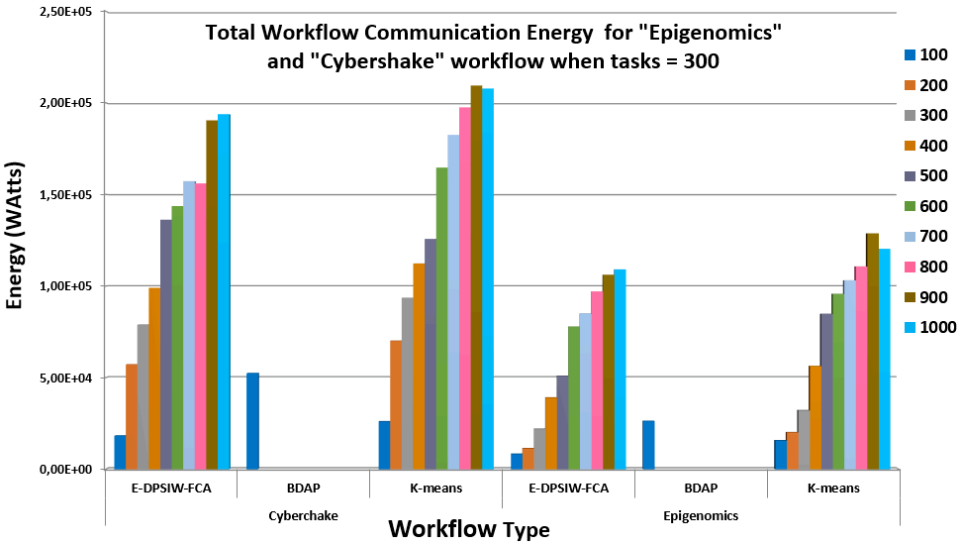


FIG. 4.7. *Total Workflow Communication energy for "Cybershake" and "Epigenomics" workflows by varying the number of VMs.*

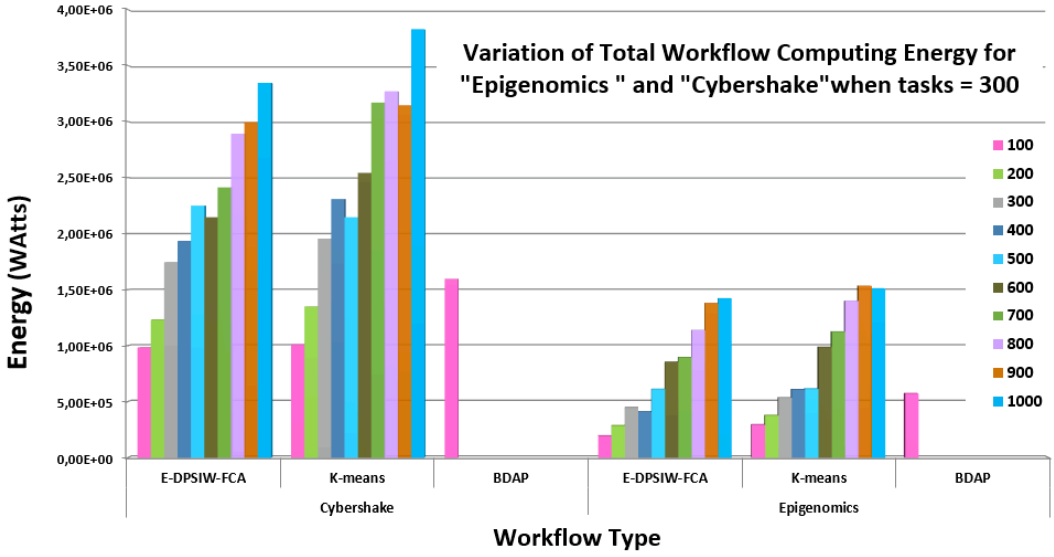


FIG. 4.8. *Workflow Computing energy for "Cybershake" and "Epigenomics" workflows by varying the number of VMs.*

(vi) *network infrastructure*- which network devices are supported (switches, routers, etc.).

From Table 5.1, we can make the following observations:

(i) Most approaches (72.72%) focus mainly in reducing data movement. Nevertheless, non-one of the above-mentioned approaches do consider the energy consumption of network devices and communication links during the workflow execution.

(ii) The vast majority of the presented works did not provide information about the network infrastructure (68.18%). However, only (4.54%) operates at the lowest level of network data center and consider its communication levels during data placement.

(iii) 22.72% of works did not provide information about data center infrastructure. Contrarily, few approaches (13.63%) ([14], [26] and [27]) considered the granularity of the data center resources, and solve the

TABLE 5.1
Comparative Table of Existing Data Placement Strategies

Approach	Objective	Technique Applied	Modelization	Datasets Type	Data Center Infrastructure	Network Infrastructure
[14]	data movement communication cost	GA	Matrix	Inp, Int Fixd	VMs	-
[16]	number of data transfers	GA	Matrix	Inp, Int	Data center	-
[17]	execution time data transfer time	GA	Matrix	Inp	Data center	Bandwidth
[18]	data transmissions data transfer time	GA	Graph	Inp, Int	Data center	Bandwidth
[19]	data transfer number	Hierarchical partionning Clustering Algorithm + PSO	Matrix	Inp, Fix, Flex	Data center	-
[20]	amount of transferred data	PSO + BEA	Matrix	Inp, Fix, Int	Data center	-
[21]	data movements number	PSO based on GA	Graph	Inp, Int	Data center	-
[22]	data transfer cost	Discrete PSO (DPSO)	Graph	Inp, Int Fix, Flex	Data center	Bandwidth
[23]	transmission time	PSO + GA	Graph	Inp, Int	Data center	Bandwidth
[24]	data security data transfer time	ACO	-	Inp, Int, Fix, Flex	Data center	-
[4]	data movement	K-means + Recursive partitioning	Matrix	Inp, Int Fix, Flex	Data center	-
[25]	total data scheduling	-	Matrix	Inp, Int	Data center	-
[26]	data movement	BEA	Matrix	Inp	VMs	-
[27]	data movement time consumption	heuristic tree-to-tree	-	Inp, Fix	Servers	Bandwidth
[27]	overall data access cost	subgradient optimization heuristics algorithm	-	Inp	Data center	Bandwidth
[29]	execution time, data movement	BEA	Matrix	Inp	Data center	-
[30]	Data transfer time	heuristic GA	Matrix	Inp	Data center	-
[31]	average query span	HPA	Hypergraph	-	Data center	-
[32]	Total amount of file transfers	HPA	Hypergraph	Inp	Data center	-
[33]	data movement	Data Correlation + BEA	Matrix	Inp	Data center	-
[3]	average query spans Execution time	FCA	Formal Context	Inp	Data center	-

data placement problem at the VM level.

Overall, to remedy these shortcomings cited above, we propose in this scope an extension of our previous work [12] that makes allowances for:

(i) The placement of datasets into SC while considering the energy consumption in computing, storage and communication devices.
(ii) Taking into account the architecture and the characteristics of the physical network interconnecting SC.
(iii) The communication between SC, while examining all the communication levels of data center architecture.

6. Conclusion. In this paper, a data placement strategy based on the FCA approach (E-DPSIW-FCA) is proposed to operate within the context of data intensive workflows. E-DPSIW-FCA minimized the total workflow communication and computing energy consumption and the total data movement between SC during the execution of the workflow tasks. Our experiments based on three types of scientific workflows showed that our strategy outperformed other strategies on reducing greatly the data movement and both computing and communication energy consumption as well as the data transfer cost. In our solution, we have considered the data placement for executing a single workflow. However, in the real world, multiple workflows can be executed concurrently. Thus, as future work, we plan to extend our strategy to consider the data placement of multiple workflows simultaneously, which is recognized today as an important challenging issue.

REFERENCES

[1] E. Deelman and A. Chervenak, Data Management Challenges of Data-Intensive Scientific Workflows. *Eighth IEEE International Symposium on Cluster Computing and the Grid (CCGRID)*, pp. 687-692, 2008.

[2] Sh. Zhang, Ch. Zhu, J. K. O. Sin and P. K. T. Mok, A Novel Ultrathin Elevated Channel Low-temperature PolySi TFT. *IEEE Electron Device Letters*, **20**, vol. 20, no. 11, pp. 569-571, 1999.

[3] K. Bousselmi, Z. Brahmi and M.M. Gammoudi, QoS-aware scheduling of Workflows in Cloud Computing environments. *2016 IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, pp. 737-745, 2016.

[4] D. Yuan, Y. Yang, X. Liu and J. Chen, A Data Placement Strategy in Scientific Cloud Workflows, *Future Generation Computer Systems*, **26**, 1200-1214.

[5] https://scec.usc.edu/CyberShake/. [Jan. 14, 2019].

[6] https://pegasus.isi.edu/. [Fev. 11, 2019].

[7] https://cloud.netapp.com/blog/ebs-efs-amazons3-best-cloud-storage-system/. [Sep. 27, 2018].

[8] https://aws.amazon.com/ebs/. [Oct. 15, 2018].

[9] https://www.ibm.com/cloud-computing/bluemix/fr/block-storage/. [Oct. 24, 2018].

[10] J. M. Pierson, Large-scale Distributed Systems and Energy Efficiency. A Holistic View. *John Wiley and Sons*, 2015.

[11] T. Li, W. Yang and A. Y. Zomaya, An energy-efficient virtual machine placement and route scheduling scheme in data center networks. *Future Generation Computer Systems*, **77**, 1-11, 2017.

[12] N. Cordeschi, M. Shojafar and E. Baccarelli, Energy-saving self-configuring networked data centers. *Computer Networks*, **57**, 3479-3491, 2013.

[13] Q. Li, K. Wang, S. Wei, L. Xu and M. Gao, A data placement strategy based on clustering and consistent hashing algorithm in Cloud Computing. *9th International Conference on Communications and Networking*, pp. 478-483, 2014.

[14] M. Ebrahimi, A. Mohan, A. Kashlev and S. Lu, BDAP. A Big Data Placement Strategy for Cloud-Based Scientific Workflows. *IEEE First International Conference on Big Data Computing Service and Applications*, pp. 105-114, 2015.

[15] Z. Brahmi, S. Mili and R. Derouiche, Data Placement Strategy for Massive Data Applications base on FCA Approach. *IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, pp. 1-8, 2016.

[16] Q. Xu, Z. Xu and T. Wang, A Data-Placement Strategy Based on Genetic Algorithm in Cloud Computing. *International Journal of Intelligence Science*, **05**, 145-157, 2015.

[17] J. Taheri and A.Y. Zomaya, Genetic algorithm in finding Pareto frontier of optimizing data transfer versus job execution in grids. *IEEE 26th International Parallel and Distributed Processing Symposium Workshops and PhD Forum*, pp. 2130-2139, 2012.

[18] L. Cui, J. Zhang, L. Yue, Y. Shi and al. A Genetic Algorithm Based Data Replica Placement Strategy for Scientific Applications in Clouds. *IEEE Transactions on Services Computing*, **11**, 727-739, 2018.

[19] Z. Er-Dun, Q. Yong-Qiang, X. Xing-Xing and C. Yi, A Data Placement Strategy Based on Genetic Algorithm for Scientific Workflows. *Eighth International Conference on Computational Intelligence and Security*, pp. 146-149, 2012.

[20] Q. Zhao, C. Xiong, K. Zhang, Y. Yue and J. Yang, A Data Placement Algorithm for Data Intensive Application in Cloud. *International Journal of Grid and Distributed Computing*, **9**, 145-156, 2016.

[21] Z. Xiang, T. Liu, B. Lin and al. A Data Placement Strategy for Scientific Workflow in Hybrid Cloud. *IEEE 11th International Conference on Cloud Computing*, pp. 556-563, 2018.

[22] X. Li, L. Zhang, Y. Wu, and al. A Novel Workow-Level Data Placement Strategy for Data-Sharing Scientic Cloud Workows. *IEEE Transactions on Services Computing*, 1-1, 2016.

[23] B. Lin, F. Zhu, J. Chen, X. Chen, N. N. Xiong and L. J. Mauri, A Time-driven Data Placement Strategy for a Scientific Workflow Combining Edge Computing and Cloud Computing. *CoRR*, 2019.

[24] W. Lei, S. Peng, W. Du, W. Wang, and G. S. Zeng, Security-aware intermediate data placement strategy in scientific cloud workflows. *Knowledge and Information Systems*, **41**, 423-447, 2014.

[25] T. Wang, S. Yao, Z. Xu and S. Jia, DCCP: an effective data placement strategy for data-intensive computations in distributed cloud computing systems. *Journal of Supercomputing*, **72**, 2537-2564, 2016.

[26] L. Liu, J. Song, H. Wang, and P. Lv, BRPS: A Big Data Placement Strategy for Data Intensive Applications. *IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pp.813-820, 2016.

[27] Q. Zhao, C. Xiong, and P. Wang, Heuristic Data Placement for Data Intensive Applications in Heterogeneous Cloud. *Journal of Electrical and Computer Engineering*, **2016**, 8, 2016.

[28] J. Zhang, J. Chen, J. Luo, and A. Song , Efficient location-aware data placement for data-intensive applications in geo-distributed scientific data centers. *Tsinghua Science and Technology*, **21**, 47181, 2016.

[29] H. Kim, Y. and Kim, An adaptive data placement strategy in scientific workflows over cloud computing environments. *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pp. 1-5, 2018.

[30] X. Qiang, Xu. Zhengquan and W. Tao A Data-Placement Strategy Based on Genetic Algorithm in Cloud Computing. *International Journal of Intelligence Science*. **05**, 145-157, 2015.

[31] A. K. Kayyoor, A. Deshpande and S. Khuller, Data placement and replica selection for improving co-location in distributed environments. 1302-4168, 2013.

[32] mit. V. çatalyürek, K. Kaya and B. Uçar, Integrated data placement and task assignment for scientific workflows in clouds. *In: Proceedings of the Fourth International Workshop on Data-intensive Distributed Computing*, ACM, 2011.

[33] Q. Zhao, C. Xiong, X. Zhao, C. Yu and J. Xiao, A Data Placement Strategy for Data-Intensive Scientific Workflows in Cloud. *15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pp. 928-934, 2015.

[34] https://aws.amazon.com/s3/fags/?nc1=h-ls/. [Nov. 19, 2018].

[35] https://www.cisco.com/. [Nov. 25, 2018]
[36] X. Meng, V. Pappas and L. Zhang, Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement. *2010 Proceedings IEEE INFOCOM*, pp. 1-9, 2010.
[37] R.L. Graham, D.E. Knuth and O. Patashnik, Integer Functions, Ch. 3 in *Concrete Mathematics: A Foundation for Computer Science*, MA: Addison-Wesley, 1994.
[38] F. P. Tso, K. Oikonomou, E. Kavvadia and al. , S-CORE: Scalable Communication Cost Reduction in Data Center Environments. *School of Computing Science*, University of Glasgow, 2013.
[39] G. Stumme, R. Taouil, Y. Bastide and al., Fast computation of concept lattices using data mining techniques. *In Proceedings, 7th Int. Workshop on Knowledge Representation Meets Databases (KRDB 2000)*, 2000.
[40] L. Lakhal and G. Stumme , Efficient mining of association rules based on formal concept analysis. *Formal Concept Analysis Foundations and Applications*, 2005.
[41] C. Carpineto and C. Romano, A lattice conceptual clustering system and its application to browsing retrieval. *Machine Learning*, **24**, 95122, 1996.
[42] B. Ganter and R. Wille, Formal Concept Analysis: Mathematical Foundations. 1999.
[43] R. Godin, R. Missaoui and H. Alaoui, Incremental concept formation algorithms based on Galois lattices. *Computation Intelligence*, **11**, 246267, 1995.
[44] E. M. Norris, An algorithm for computing the maximal rectangles in a binary relation. *Revue Roumaine de Mathmatiques Pures et Appliques*, **23**, 243250, 1978.
[45] G.Bernhard and W. Rudolf, Formal concept analysis. *Mathematical foundations*, 2012.
[46] S. O. Kuznetsov, A fast algorithm for computing all intersections of objects in a nite semilattice. *Automatic Documentation and Mathematical Linguistics*, **27**, 1121, 1993.
[47] J. P. Bordat, Calcul pratique du treillis de Galois dune correspondance. *Math. Sci. Hum.*, **96**, 31 47, 1986.
[48] Pegasus workow gallery: https://pegasus.isi.edu/ workow gallery/. [Fev. 21, 2019]
[49] https://scec.usc.edu/scecpedia/ CyberShake/. [Fev. 21, 2019]
[50] http://montage.ipac.caltech.edu/. [Fev. 21, 2019]
[51] http://epigenome.usc.edu/. [Fev. 21, 2019]
[52] L. Nourine, and O. Raynaud, Fast algorithm for building Lattices. *Information Processing Letters*, **71**, 199-204, 1999.
[53] https://github.com/manoelcampos/cloudsim-plus. [Fev. 11, 2018]
[54] I. T. Cotes-Ruiz, R. P. Prado, S. García-Galán, J. E. Muñoz-Expósito and N. Ruiz-Reyes, Dynamic Voltage Frequency Scaling Simulator for Real Workflows Energy-Aware Management. *in Green Cloud Computing*, 2017.
[55] M. Shojafar, C. Canali and R. Lancellott, Adaptive Computing-plus-Communication Optimization Framework for Multimedia Processing in Cloud Systems. *IEEE Transactions on Cloud Computing*, 2016.
[56] C. Fiandrino, D. Kliazovich, P. Bouvry and al., Performance Metrics for Data Center Communication Systems. *2015 IEEE 8th International Conference on Cloud Computing*, 2015.
[57] T. Guérout, T. Monteil, G. D. Costa and al., Energy-aware simulation with DVFS. 2013.
[58] https://azure.microsoft.com/en-us/pricing/details/bandwidth/. [Mar. 15, 2019]
[59] https://searchaws.techtarget.com/definition/availability-zones. [Apr. 3, 2019]
[60] M. Dias. Sérgio and José. Vieira Newton, Reducing the Size of Concept Lattices: The JBOS Approach. 2010.
[61] S. Kuznetsov and S. Obiedkov, Comparing performance of algorithms for generating concept lattices. *J. Exp. Theor. Artif. Intell*, **14**, 189-216, 2002.
[62] AK. Jain, Data clustering: 50 years beyond K-means. *Pattern recognition letters*, **31(8)**, 651-666, 2010.
[63] K. Bousselmi, Z. Brahmi and M. M. Gammoudi, Energy Efficient Partitioning and Scheduling Approach for Scientific Workflows in the Cloud.*IEEE International Conference on Services Computing (SCC)*, 2016.

Edited by: Dana Petcu
Received: April 22, 2019
Accepted: June 18, 2019

DOI 10.12694/scpe.v20i3.1572
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 3, pp. 563–576. http://www.scpe.org

SIMD IMPLEMENTATION OF THE AHO-CORASICK ALGORITHM USING INTEL AVX2

OURLIS LAZHAR *AND BELLALA DJAMEL †

Abstract. The Aho-Corasick (AC) algorithm is a multiple pattern exact string-matching algorithm proposed by Alfred V. Aho and Margaret J. Corasick. It is used to locate all occurrences of a finite set of patterns within an input text simultaneously. The AC algorithm is in the heart of many applications including digital forensics such as digital signatures demonstrating the authenticity of a digital message or document, full text search (utility programs such as *grept*, *awk* and *sed* of Unix systems), information retrieval (biological sequence analysis and gene identification), intrusion detection systems (IDS) in computer networks like SNORT, web filtering, spam filters, and anti-malware solutions (virus scanner). In this paper we present a vectorized version of the AC algorithm designed with the use of packed instructions based on the Intel®streaming SIMD (Single Instruction Multiple Data) extensions AVX2 (Advanced Vector Extensions 2.0). This paper shows that the vectorized AC algorithm reduces significantly the time matching process comparing to the implementation of the original AC algorithm.

Key words: Pattern-matching, Aho-Corasick algorithm, Vectorization, Intel®Streaming SIMD Extensions 2.0 (AVX2).

AMS subject classifications. 68W32, 68W10

1. Introduction.

String-matching is the concept of finding a sequence of characters, often called patterns, inside a provided text. The matching process includes the location of these characters inside the text. Formally, pattern-matching is the problem of locating all occurrences of a pattern P of length m in a text T of length n. Both the pattern and the searched text are vectors of elements of a finite set called an alphabet Σ. We say that pattern P occurs with valid shift s in text T if $0 <= s <= n - m$ and $T[s + 1.s + m] = P[1..m]$.The pattern-matching is therefore the problem of finding all valid shifts with which a pattern P occurs inside a text T [1].

Pattern-matching algorithmic complexity is usually analysed by the running time of the algorithm and the memory space required by the computations. To this, we can add a setup time, or a substantial amount of computation before the algorithm can begin searching (time spent during the pre-processing phase), and the need for backtracking or not (since moving back and forth through the search text can entail some form of buffering if the search text doesnt exist in memory, but sent to the program as a stream of data) [2]. Considering the running time feature, all pattern-matching algorithms perform on the order of $a + bn$, where a is the pre-processing time, b is a constant indicating the number of comparisons made for each character, and n is the number of characters in the text being searched. The effectiveness of the algorithms is measured according to their ability to lower the value of b. Effective ones perform in less than one comparison per character searched; $(b < 1)$ termed as *sub-linear* pattern-matching algorithms [2].

Many algorithms to solve pattern-matching problem exist. They are classified either as single or multiple pattern algorithms based on the number of patterns to look for. Applications that relay on this class of algorithms may require exact or approximate pattern-matching [1]. Many solutions for exact string-matching of multiple patterns have been developed. However, most of them have been designed for moderately sized pattern sets, and therefore they do not fit well with larger sets of patterns. The most commonly used solutions are Aho-Corasick (AC), Wu-Manber (WM) and Commentz-Walter (CW) algorithms.

The Aho-Corasick algorithm [3] uses a set of patterns to construct a finite automaton during the pre-processing phase. The matching involves the automaton scanning the text string, stepping through the input characters one at a time and changing the state of the automaton. At every state transition for every text character, we check if there is a match by observing whether the current state is an output state (which indicates that a pattern has been found) or not. The time complexity of the AC algorithm is proportional to the total length of the patterns during the pre-processing phase and only proportional to the size of the text being processed during the searching phase [2]. The AC algorithm has the advantage of examining each text character only once, and locating all occurrences of patterns within a given text in one pass. Its major drawback is the space memory required to store the automaton states, which increases with their number.

*Department of Industrial Engineering, Faculty of Technology, University of Batna2, Batna, Algeria. (ourlisl@yahoo.fr)
†Department of Computer Science, University of Batna2, Batna, Algeria. (bellala_djamel@yahoo.co.uk).

The Wu-Manber algorithm [4] uses the bad-character shift (indicating the characters to skip during the scanning phase) from the Boyer-Moore algorithm, but looks at blocks of text instead of single characters during the scanning phase to improve the matching performance: both the pattern and the text are treated as blocks. The algorithm also builds three tables during the pre-processing phase: the hashing or SHIFT table to determine the number of characters that can be skipped when scanning the text, the HASH and PREFIX tables to determine which pattern is a candidate for the match (and eventually to verify the match) when the shift value equals to 0. In practice, the algorithm is more suitable when dealing with patterns of similar length, and its running time does not increase in proportion to the size of the pattern set. But, in case of short patterns, the scanning time increases due to the decrease in characters shifting [5].

The Commentz-Walter algorithm [6] is a suffix-based, exact multiple string matching solution that combines the concepts of Boyer-Moore and AC algorithms. In the pre-processing phase, the CW algorithm constructs a finite automaton similar to that of the AC algorithm but from the reversed set of patterns to use the shifting method of the Boyer-Moore algorithm, which usually handles mismatches in a way to obtain a sub-linear time complexity. The matching involves the automaton scanning through the text string in a backward manner: characters of the patterns are scanned from right-to-left beginning with the rightmost one, exactly as the Boyer-Moore algorithm does. The length of the matching window is the minimum pattern length. In case of a mismatch or a complete match of the pattern, the CW algorithm uses a recomputed shift table to identify portions of text to be skipped and shifts the window to the right accordingly. With small numbers of patterns to look for, Boyer-Moore aspects of the CW algorithm can make it faster than the AC algorithm, but with larger numbers of patterns, the AC algorithm has a slight advantage [7, 8].

To improve the AC algorithm, many approaches are proposed. Nishimura et al. [9]described the speed-up of string pattern-matching by collecting states used frequently for CPU (Central Processing Unit) cache efficiency using data compression. Their approach reduced the elapsed time in case of a compressed English text to about 55%. In order to accelerate their Network Intrusion Detection System (NIDS) engine, used to identify network attacks by inspecting packet content against a collection of many thousands of predefined patterns, Lin et al. [10] proposed a Graphic Processor Unit (GPU). Their algorithm on GPU achieved up to 4,000 times speed-up compared to the AC algorithm on CPU. Arudchutha et al. [11] improved the speed of the AC algorithm using a multicore CPU based software implementation through parallel manipulation of pattern-sets using POSIX (Portable Operating System Interface) thread utility to handle a large amount of data in the form of strings (Bio-computing applications). They arrived at the conclusion that their implementation gives better results compared to that of a parallel implementation of the same algorithm. In this paper, we present another technique to speed up the AC algorithm, which benefits from Intels AVX 2.0 introduced in the Haswell microarchitecture in 2013.

2. Vectorization: Data Parallelism. Vectorization, or SIMD processing, is the process in which an algorithm is converted from a sequential implementation, that performs an operation one pair of operands at a time, to a vector process where multiple data operands are simultaneously performed in only one instruction (data-level parallelism), provided that this scalar algorithm is suitable for being parallelized using vectorization techniques [12]. In SIMD processing, operands (adjacent data items of the same type and size refer to as vectors) are treated not as individual integers or float-point numbers but rather as whole vectors. Moreover, not only a single instruction operates on whole vectors but also reading from and writing to memory (load/store instructions) operate on whole vectors too.

A SIMD operation like addition, logical OR, data movement, conversion or comparison is a simple operation that is performed on vector items in parallel (all operations in SIMD are vector operations). Only one instruction is needed to process these vector items (refer to as packed data), whereas in sequential processing, without SIMD, multiple instructions would be used instead [13]. There are different approaches to achieving vectorization:

1. The first form of vectorization is to explicitly call the processors SIMD instructions using low-level assembly code and available registers. This way of utilising vector instructions offers more control over the program and yields the maximum performance of the system but cross-platform porting is quite difficult;
2. Use of the compiler intrinsic functions (assembly-coded C style functions that provide access to many Intel®instructions, including SIMD instructions, without the need to explicitly write assembly code).

In this approach, high level languages such as C/C++ or FORTRAN can be used to write the code. It provides almost the same benefits as using inline assembly and saves us to deal with register allocations, instructions scheduling and stack calls;

3. Compiler auto-vectorization: the easiest form of vectorization where no changes to source code are required. As a result, the portability of the code is preserved;
4. Use of a high-level library, such as Intel®MKL (Math Kernel library) that contains implementations which use vector instructions of common mathematical operations [14];
5. Use of Intel®Cilk™Plus array notation and elemental function syntax. These notations can be used separately from each other to help the compiler perform parallelisation. More details to using these language extensions can be found in [15].

Compiler auto-vectorization is the easiest form of vectorization, where specific portions of the sequential code are converted, by the compiler, into equivalent parallel ones intended for use on vector processors. This is the most convenient way to do vectorization because cross-platform porting is preserved in contrast to embedding SIMD assembly code into a source program, which is rather complicated because it requires careful handling of data transfers between available vector registers and memory. This latter method may deliver better performance than the compiler auto-vectorization, but cross-platform porting is not guaranteed [14]. It is chosen here because it seems to be the most effective one for speeding up the AC algorithm.

In SIMD processing, the same operation is applied to each element of the source operands. However, this is a constraint because programs often do not map well onto pure vector operations and even if it is the case, they might not reach the best resource utilisation [16]. Programs can benefit from SIMD processing if they have highly repetitive loops, and use intensively instructions that operate on independent values in parallel. The practical speed-up with vectorization comes from the efficient data movement and the identification of vectorization possibilities in the program itself [17]. Vectorization fails if the structure of the program is not suitable for parallelisation and its data types are either mixed, dependent or not aligned. Intel processors implement architectures that include data parallelism in the form of a vector instruction set. Common examples include MMX™, SSE (Streaming SIMD Extensions) and AVX instructions. In the next section, we introduce Intel®AVX used to vectorize the AC algorithm.

3. Advanced Vector Extensions (AVX). For Intel®processors, the vector instructions have been gradually introduced in different processor generations. Started with the MMX™in 1996 (instructions in this extension operate only on packed integer values and rarely used in modern processors), followed by several SSE versions from 1999 to 2008 (available in almost any today processor), and continued to this day with AVX (supported in new processors).

AVX is a 256-bit SIMD operations extension of Intel®SSE, designed to deal with applications which make a more intense use of floating-point operations (scientific applications, visual processing, data mining, cryptography, 3D modeling and gaming). In this extension, the support for integer operands is lacking. It was released in the early 2011s as part of the second-generation Intel®Core™processor family (supported first by the Intel®Sandy Bridge processor released in Q1, 2011). AVX extends the previous SIMD instructions by expanding the 128-bit SIMD registers to 256 bits, adds a three-operand nondestructive operation where the destination register is different from the two source operands, and introduces a new prefix coding scheme VEX in instruction encoding format [18]. Processors based on the Sandy Bridge microarchitecture and supporting AVX include Core i7, Core i5, and Core i3 second- and third-generation processors along with Xeon series E3, E5, and E7 processors [19].

AVX2 also known as Haswell New Instructions was released in 2013 with the fourth generation Intel®Core™ processor family (supported first by the Intel®Haswell processor released in Q2, 2013), and is designed to support integer computation demanding algorithms by extending SSE and AVX with 256-bit integer instructions. AVX2 includes the Fused Multiply-Add (FMA) extension which allows numbers to be multiplied and added in one operation, enhances vectorization with *Gather* operation that loads vector items from non-contiguous memory locations, and introduces vector Shift operations. The specification of the AVX2 instruction set and information needed to create applications using this extension are available under the Intel Architecture Instruction Set Extensions Programming Reference [20]. Processors based on the Haswell microarchitecture and supporting AVX2 include Core i7, Core i5, and Core

i3 fourth generation processors and Xeon E3 (v3) series processors (based on the Haswell microarchitecture) [19].

AVX-512 consists of multiple extensions of the AVX and AVX2 family of SIMD instructions, announced by Intel®in July of 2013. It uses a new EVEX prefix encoding with support for 512-bit vector registers, and offers a high level of compatibility with AVX.

In the vectorization process of the AC algorithm proposed next, we use AVX2 extension to benefit from SIMD instructions operating on integer data type since states generated in the pattern-matching machine by running the AC algorithm are coded as integer values.

4. Vectorization process of the Aho-Corasick algorithm. In this section, we first describe the two versions of the AC algorithm, the standard version and the version that uses the *next-move* function as proposed by the authors in [3]. Then, we present an optimized version of the AC algorithm, which is obtained through vectorizing the AC algorithm that uses the *next-move* function.

4.1. Review of the AC algorithm. The AC algorithm, first described by Alfred V. Aho and Margeret J. Corasick at Bell laboratories in 1975 [3], is a direct extension of the Knuth-Morris-Pratt (KMP) algorithm [21], which uses automata approach to solve multiple pattern-matching problems. Initially designed to accelerate the library bibliographic search program. The performance gained using this algorithm was between 5x and 10x faster compared to the original program [3].

The AC algorithm builds a finite state machine (FSM) from patterns, then uses the pattern-matching machine to locate all occurrences of patterns (that may overlap) within a given text in one pass. Each state of the machine is identified by a number (an integer value). When a character of an input text is processed, one or more finite automaton state transitions are made. State transitions (or machine behaviour) are dictated by three functions, namely: (i) the *goto* function g indicates state transitions by mapping a pair (current state, input character) into either a state or a fail state. In the latter case, the fail state is indicated by calling the failure function. Some of these states are designed as output states indicating that a set of patterns has been found, (ii) the *failure* function f defines which state transition to move to in case of a mismatched character, (iii) the *output* function indicates the patterns found and their locations in the text when the machine reaches an output state.

We can better describe how the algorithm works by considering an example. Suppose we want to search a text for the set of patterns {these, this, the, set}. An appropriate state machine for these four (04) patterns is described in the Goto, Failure and Output tables as shown in Fig 4.1. We start with an empty *goto* function (*all characters* $\rightarrow S0$), and then we add all patterns, one by one. In the same time, we construct the Output table. In the second step, we compute the *failure* function and update the Output table as described next.

Once the automaton is built, the matching process is straightforward using the above-mentioned functions. The machine inspects all characters from the beginning of the text successively, one character at a time, by changing the state of the automaton (*goto* function is operating) and occasionally emitting output (*output* function is operating). If there is no valid state transition for the character under inspection, then the machine detects a transition failure and tries to investigate a match from other states (*failure* function is operating). We start in state 0, and then we examine each character of the text. For example, if a character t is seen in state 0, then we move to state 1. An h character would then take us to state 2. However, if the next character is not an e or i, we consult Fail [2] and change to state 0. Note that from state 2, Goto[2] ['e'] = 3 and Goto[2] ['i'] = 6, but Goto [2][anything else] = FAIL_STATE. If we arrive at a state with a non-empty Output function, then a matching string is found. The corresponding c++ code is given in listing 1.

Listing 1
C++ implementation of algorithm 1 - Pattern-matching maching according to [3]

```
//str is a pointer to the text to search
//N is the number of the characters to be examined (size of the text)

void AC_Search(unsigned char *str, int N)
{
   int state = 0;
   for (int i = 0; i<N; i++)
   {
```

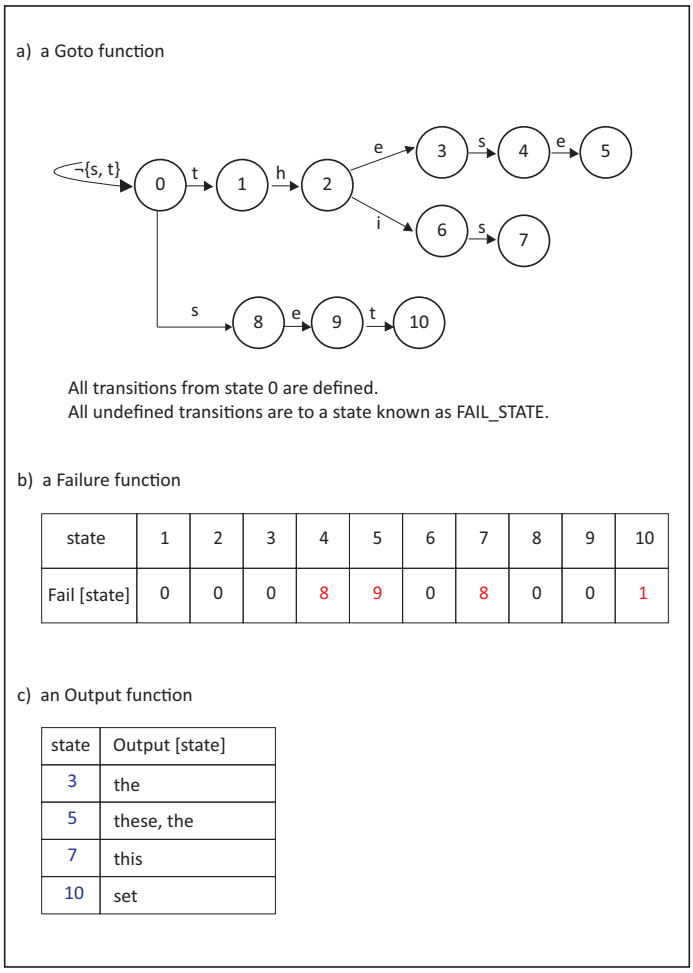


FIG. 4.1. *Pattern-matching machine for the set of patterns {these, this, the, set}*

```
        unsigned char c = str[i];
        while(Goto[state][c] == FAIL_STATE) {state = Fail[state];}
        state = Goto[state][c];
        //output
         if (Output[state] != NULL) {//Here we report a match}
    }
}
```

In the following, we construct the various tables of the AC algorithm. First, we construct the Goto and Output tables. We start with an empty Goto table and step by step we process each word from the pattern set by keeping track of the state transition made in the Goto table when examining each character. Where possible, we overlay patterns that begin with the same characters. We also construct the Output table at the same time: each pattern is identified within the Output table by using its output state as an index to this table. For example, to build the Goto table in Fig 4.1, we start with the pattern *these*($S0 - t \rightarrow S1 - h \rightarrow S2 - e \rightarrow S3 - s \rightarrow S4 - e \rightarrow S5$). Next, we add the pattern *this*($S2 - i \rightarrow S6 - s \rightarrow S7$), and finally the pattern *set*($S0 - s \rightarrow S8 - e \rightarrow S9 - t \rightarrow S10$). The pattern *the*($S0 - t \rightarrow S1 - h \rightarrow S2 - e \rightarrow S3$) overlays with the pattern *these*, which is already processed. We just consider its output state in the Output table. Then, we construct the Fail table by taking into account the overlapping search patterns, and the possibility that a given state might find multiple search patterns. We proceed by examining all states of the Goto table by defining "depth" to be the distance of state x from state 0. Then, states 1 and 8 are at depth 1, while states 2 and 9

are at depth 2. Intuitively, Fail[depth 1 states] should bring us to state 0. But Fail [depth 2 states] depends on both Fail and Goto functions of the depth 0 and 1 states. In general terms, Fail[depth n states] depends on the Fail and Goto functions of all states of lesser depth. Consider, for example, the computation of Fail [10] shown in Fig 4.1. The character that brought us from state 3 to state 4 was *s*. Now, we search all other transitions from state 0 to state 3 that make use of *s*. We find just one *s* that takes us from state 0 to state 8. Therefore, Fail [10] is set to 8. Note how the calculation of Fail has turned up the overlap of the patterns *these* and *the*. Output is updated at the same time. When an overlap is discovered, we need only to update (by concatenating) the content indexed by the two overlapping states.

Finally, we must deal with issues that may affect the performance of the algorithm such as its searching speed and the memory requirements. Indeed, we must define the number of possible states, stored as integers, that our machine might have *(#define MAXSTATES xx)*. This number depends on the number and the length of the patterns to look for. For example, in case of searching an English text for a set of 10 short patterns (words) having length less or equal 18 characters, MAXSTATES is set to about 80. While searching for a set of 10 long patterns (sentences) having length between 19 and 70 characters, MAXSTATES will be set to about 600 (see the experience results reported in Tables 5.2 and 5.3). The next issue is how to store the state transitions of the Goto function. This would involve a jump table for each state, which depends on the type of the search being performed: if we are searching an English text for some words (patterns are subset of 256 characters), then the table would contain a next state for each ASCII character (among 256 possibilities), and the Goto table will be defined as follows *(int Goto[MAXSTATES][256];)*. While searching DNA sequences for DNA keywords (patterns are subset of 4 characters A, C, G and T), the Goto table will be declared as follows *(int Goto[MAXSTATES][4];)*. Finally, when dealing with virus signatures based on byte stream (patterns are subset of hexadecimal characters and wildcards), defining the Goto table as *(int Goto[MAXSTATES][25];)* is more than enough. Notice that, the memory requirements of the AC algorithm can be taken directly from the different tables used in constructing the automaton during the pre-processing phase since it is the only structure used in the matching process. Unfortunately, the space complexity can be quite large depending on the alphabet and the patterns set. In the worst case it would be O(mc) where c is the size of the alphabet Σ and m is the total length of the patterns.

4.2. Review of the AC algorithm using the *next-move* function. The failure function f as implemented in the previous AC algorithm is not optimal. Consider the pattern-matching machine of Fig 4.1. We see Goto[10] ['e'] = 8. If the machine is in state 4 and the current input character is not an *e*, then the machine would enter state Fail[10] = 8. Since the machine has already determined that an input character is not an *e*, it does not need to consider the value of the goto function of state 8 on *e*. In fact, if the pattern *set* was not present, then the machine could change directly from state 4 to state 0. Thus, skipping an intermediate transition to state 0. To eliminate unnecessary failure transitions in the previous version of the AC algorithm (the standard version), the *next-move* function of a deterministic finite automaton (DFA) can be used in place of the goto and failure functions. Converting the algorithm to a DFA allows to indicate for each pair (given state, given character) the next state to move to, which is always a valid state.

The *next-move* function is computed from the goto and failure functions using algorithm 4 in [3]. The purpose of this function is to build the three (03) following tables: len_movef, movef_char and movef_nextstate (indicating respectively for each state the number of characters to process, the character being processed and the next state in case of a match) used by the algorithm. The *next-move* function is coded below as follows. For example, in state 0, we have a transition on *s* to state 8, a transition on *t* to state 1, and all other transitions on any other characters are to state 0 (not shown in Fig 4.2). The c++ code for the *next-move* function is given in listing 2.

Listing 2
C++ implementation of the AC algorithm using the next-move function according to [3]

```
//len_movef:table indicating the number of characters to process for each state
//movef_char: table indicating the character being processed for a given state
//movef_nextstate: table indicating the next state if a match occurs

void AC_Nextmove_search(unsigned char *str, int N)
{
```

```
   int state = 0;
   for (int j = 0; j<N; j++)
   {
      unsigned char c = str[j];
      for (int i = 0; i < len_movef[state]; i++)
   {
      if (c == movef_char[state][i])
      {
         state = movef_nextstate[state][i];
         goto nextOK;
      }
   }
   state = 0;
   nextOK:

   //Statement used in [3] and can be skipped if we want to do output later
   if (Output[state] != NULL) { //Here we report a match }

   //Here we use an array to do output after processing the whole text
   OutCount[state] += OutPutf[state]
   }
}
```

Theoretically, the DFA can reduce up to 50% of state transitions, which would reduce extra comparisons and accelerate significantly the matching process. In practice such accelerations cannot be reached since the pattern-matching machine spends a lot of its time in state 0, from which there are no failure transitions [3] as shown in our experience results reported in Tables 5.1, 5.2 and 5.3.

From the above pseudo-code and algorithms 1, 2 and 3 given in [3], it appears clearly that building the different tables is the difficult task of the AC algorithm. The searching process is easy once these tables are available. This method of encoding state transitions is more economical than storing them as a two-dimensional array. However, it requires extra memory space larger than the corresponding representation for the goto function. Using the *next-move* function with the set of patterns {these, this, the, set} would make the sequence of state transitions shown in Fig 4.2.

4.3. Simultaneous searching of multiple patterns using AVX2. One effective way to benefit from different levels of parallelism included in modern processors is the use of vectorization. The program to optimise has to be written using available vector registers (XMM, YMM and ZMM in case of Intel®processors) and SIMD instructions (MMX^{TM}, Streaming SIMD Extensions SSE and AVX instructions in case of Intel®processors).

In this section, we try to identify vectorization opportunities from the structure of the AC program (its code structure), we particularly look to parallelise some data processing. The second version of the AC algorithm, using the *next-move* function, seems vectorizable. Considering the sequence of state transitions in Fig 4.2. Each time the machine enters a new state, a text character being inspected, has to be matched against all characters of that state transitions one at a time. One way to speed up the searching process is by comparing simultaneously the text character to all characters of that state transitions. For example, if the machine enters state S2 then the character being examined is concurrently compared to e, i, t and s characters (data-level parallelism). This can be achieved using vector processing techniques.

To implement the vectorized algorithm, we use VPBROADCASTB and VPCMPEQB Intel®AVX2 instructions. The VPBROADCASTB instruction loads a byte integer (a text character being processed) from memory via ECX and EDX registers and broadcasts it to thirty-two (32) locations in YMM1 register (here we are coding in 32-bit mode). Next, we load all characters of state transitions of a current state in YMM2 register using VMOVUPS instruction. Finally, a SIMD comparison is performed between these two registers YMM1 and YMM2 using VPCMPEQB instruction, and the result is given in YMM3 register. The next state is indicated by using the value of YMM3 register as an index to another table, namely movef_nextstate table. This parallel matching process, as shown in Fig 4.3, is repeated until all characters of an input text are processed.

Note that when the number of patterns of a given state (max = 32 patterns/state) is not important, only few bytes of YMM2 register will be used. In this case, as shown in the example of Fig 4.3, we can write code to avoid comparing YMM1 register bytes ranging from byte 4 to byte 31 (having "ch$_i$" value) to their counterparts in YMM2 register (having zero value), but this is useless since the speed of comparison will not be affected

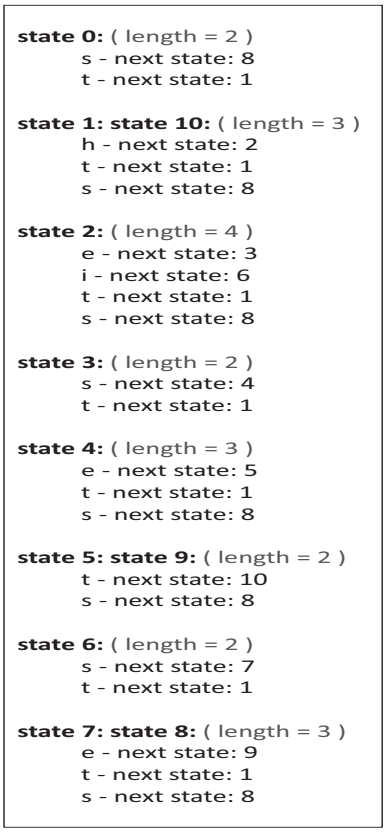


FIG. 4.2. *State transitions using the next-move function*

(remains the same) because we are doing SIMD comparison. The vectorized version of the AC program using the *next-move* function is given in listing 3.

LISTING 3
AVX2 implementation of the AC algorithm using the next-move function

```
// Here we convert the scalar AC algorithm in listing 2 to to a vector process

_declspec(naked) void __fastcall AC_NextMove_search_avx2(unsigned char *strp,int N)
{
    // fastcall ==>     edx == N       ecx == strp    eax   ==  last param
   _asm push ebp
   _asm mov ebp, esp
   _asm push esi
   _asm push ebx
   _asm push edi

   _asm  add edx, ecx  // edx == N+strp
   _asm  xor esi, esi  // esi == state  == 0

 loopj :
   _asm VPBROADCASTB ymm1, [ecx]      // ymm1 size = 32 bytes(characters)
   // first we start by coding the loopi code fragment in listing 2:
   // (1) for (int i = 0; i < len_movef[state]; i++) { ......}
   // (2) if (c == movef_char[state][i]) { ..(3)....}
   _asm lea ebx, movef_char
   _asm mov eax, esi
   _asm shl eax, 8   //edx == state*256 (256 bytes-line size in "movef_char" array)
   _asm vmovups ymm2, [eax + ebx]
   _asm vpcmpeqb ymm3, ymm2, ymm1
            //comparison result in ymm3   0xff 0x00 0x00 0xff .....  0xff 0x00
   _asm vpmovmskb eax, ymm3
            //comparison result in eax      1    0    0    1  .....    1    0
```

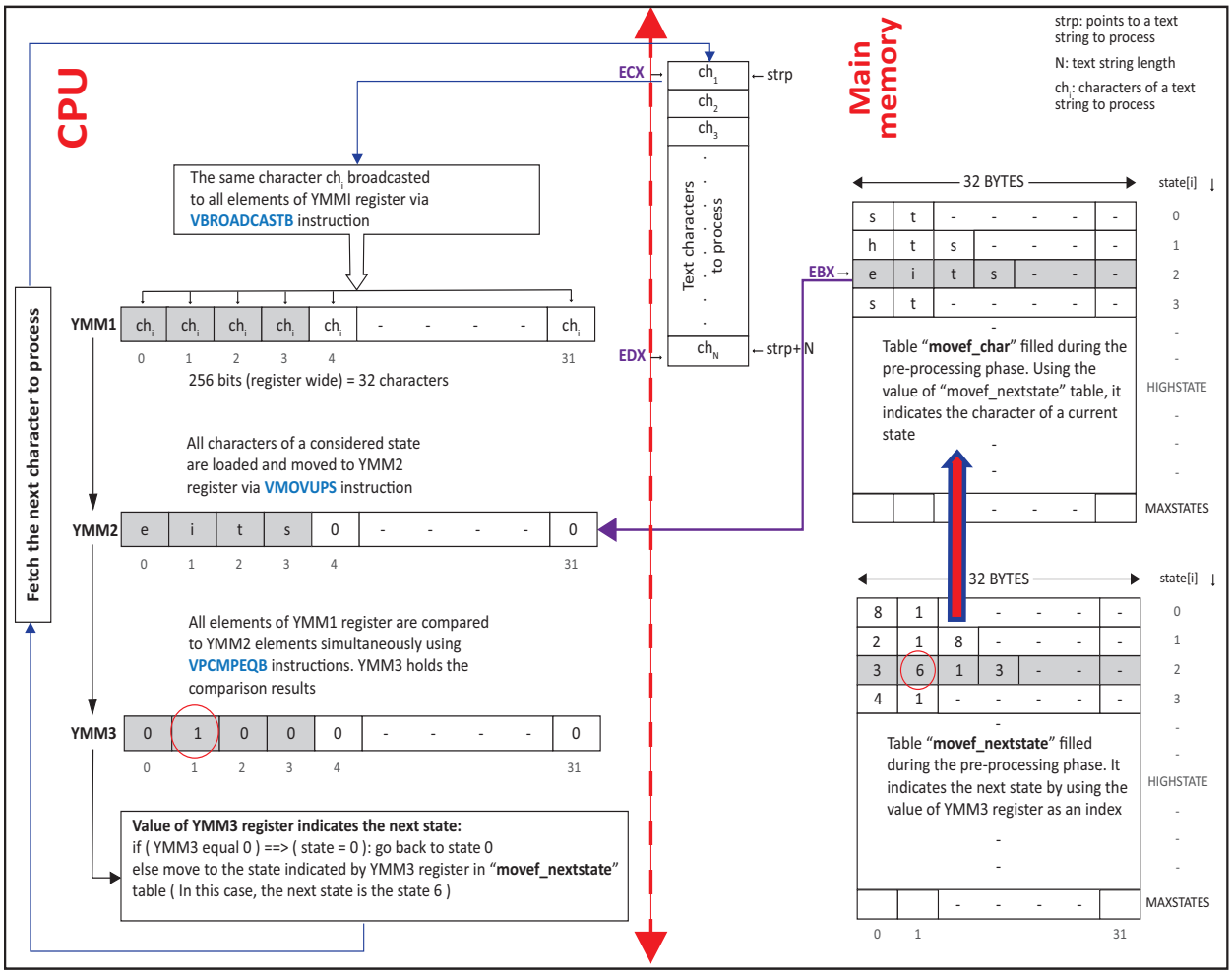


FIG. 4.3. *Vectorization process of the AC algorithm using AVX2 instructions*

```
  _asm bsf eax, eax //i == eax  ==                                    2

  //for a given state, the number of state transitions <=32 (len_movef[state]<=32)
  _asm jz  nonzero                 // bsf instruction (if flagzero == 1)  then jump
      // (3) {state = movef_nextstate[state][i]; goto nextOK;}
      // state  = state*256, 256 bytes - size of a line in "movef_nextstate" array
  _asm shl esi, 8
  _asm add esi, eax               //state  = state + i
   _asm lea edi, movef_nextstate //state = movef_nextstate[state][i];
  _asm mov esi, [edi + esi*4]     //4 bytes-"movef_nextstate" array element size
  goto nextOK;

nonzero:
  _asm xor esi,esi

nextOK :  // OutCount[state] += OutPutf[state];
  _asm mov eax, OutPutf[  esi * 4]    //OutPutf must be static
  _asm add OutCount[esi * 4],  eax    //OutCount must be static

  _asm inc ecx
  _asm cmp ecx,edx
  _asm jle loopj

  _asm pop edi
  _asm pop ebx
```

```
    _asm pop esi
    _asm pop ebp
    _asm ret
}
```

5. Experimental results. In this section we evaluate the performance of the vectorized AC algorithm by comparing it to the AC algorithm using the *next-move* function as proposed by the authors in [3]. In particular, we compare the running time of the AC_NextMove_search_AVX2 program to the AC_NextMove_search _ASM program (the AC algorithm using the *next-move* function before vectorizing it coded in assembly, not listed in this paper). Comparison is done by including the time of the pre-processing phase.

The programs have been compiled with Microsoft Visual Studio Ultimate 2013 (version 12.0.21005.1 REL) and experiments were executed on LENOVO laptop, a 1.9 GHz CPU Intel I3-4030U with 4 GB RAM running Microsoft Windows 8.1 Pro 64 bits.

For the evaluation purpose, we first use different files of plain English text, collected from the Gutenberg website (`http://www.gutenberg.org`), their total size is ranging from 1 MB to 1 GB, and different pattern size categorised as follows: (i) *sp* short patterns designing English words of length less or equal 18 bytes and, (ii) *lp* long patterns designing English phrases of length greater than 18 bytes. Patterns were arbitrarily chosen, some of them may exist or not in text files to be processed (especially when matching an important number of patterns).

Each input file is processed by running respectively AC_NextMove_search_ASM and AC_NextMove _search_AVX2 programs. During the matching phase, the text characters of each file is matched against either *sp* and *lp* patterns of different size. All functions of the pre-processing phase are coded in C++ language. Only the search function, which behaves as a finite state string pattern-matching machine is coded in assembly (in case of AC_NextMove_search_ASM program) and codded using AVX2 instructions (in AC_NextMove_search_AVX2 program). The pre-processing phase consists of the following functions: the *goto*, the *failure* and the *next-move* functions that implement respectively algorithms 2, 3 and 4 in [3].

Table 5.1 summarises the running times of the algorithms. Here we would point out that the vectorized AC algorithm, as implemented here, use only one YMM register (32 bytes length), which limits the number of state transitions to 32 (that can be reached once more than 100 patterns are used especially in case of long ones).

To deal with an important number of patterns, the AC_NextMove_AVX2 program has to be adjusted in a way that it uses a variable to store YMM value when the number of next states (state transitions) exceeds 32, or uses more than one YMM register instead. Another alternative is the use of ZMM registers (AVX- 512) where the width of the register is increased to 512 bits allowing to have up to 64 state transitions/register. These proposals will be investigated in depth in a future work.

The experimental results reported in Table 5.1 show that the AC_NextMove_AVX2 program performs better than its counterpart AC_NextMove_ASM in all cases. The speed up gained depends on the patterns length and the number of patterns to search for.

It is important to highlight the fact that in practice the AC algorithm, as stated by the authors in [3] and confirmed by our experiments, spends most of its time in state 0 (more than 90% of its processing time in most cases). From Table 5.1, we can also notice that the speed-up gained is independent on the file size.

In the second phase of our experiment, we use 10 different patterns (*sp* first then *lp* next having different lengths) to search for, through different file sizes, and we measure the speed-up gained in terms of time processing (in %) when executing the vectorized AC algorithm (see Tables 5.2 and 5.3). Here, we give a theoretical value (TheoV) of the speed-up when comparing the running time of the A_NextMove_AVX2 algorithm against the AC_NextMove_ASM algorithm. This value is calculated by incrementing a variable inside an inner loop in AC_NextMove_search function and dividing its value by the number of the processed characters at the end of the outer loop (which is the size of the input file). When considering these values, it is clearly noticeable from Tables 5.2 and 5.3 that the speed-up gained is independent on the file size.

Table 5.4 shows the differences, from the results of experiments, between our implementation of the AC algorithm and some of the implementations discussed here.

6. The vectorized AC code portability. As mentioned earlier, the vectorized AC code consists up of the following functions:

TABLE 5.1
Running times obtained while searching different text file size for sets of different patterns

Text file size (KB)	Number of patterns	Total patterns length (Bytes)	T1(sec) AC NextMove (ASM)	T2(sec) AC NextMove (AVX2)	Speed-up(%) [AVX2/ ASM]	Time spent in state 0 (%)
1125 (1MB)	1 sp	8	0,005340	0,004020	32,84	99,43
	10 sp	53	0,014807	0,005675	160,92	96,90
	100 sp	632	0,043139	0,016071	168,43	76,96
	1 lp	55	0,004349	0,002532	71,76	99,80
	10 lp	578	0,013965	0,004723	195,68	98,95
	100 lp	3914	0,043059	0,018160	137,11	74,27
9690 (10MB)	1 sp	8	0,036198	0,022953	57,70	99,46
	10 sp	53	0,130331	0,054303	140,01	97,01
	100 sp	632	0,360853	0,142437	153,34	76,94
	1 lp	55	0,041455	0,024482	69,33	99,77
	10 lp	578	0,125042	0,044071	183,73	98,76
	100 lp	3914	0,303146	0,151035	100,71	76,07
102240 (100MB)	1 sp	8	0,479668	0,368961	30,01	99,49
	10 sp	53	1,444595	0,592323	143,89	97,25
	100 sp	632	4,546556	1,694503	168,31	77,31
	1 lp	55	0,385994	0,193441	99,54	99,75
	10 lp	578	1,399632	0,509368	174,78	98,99
	100 lp	3914	3,659735	1,775760	106,09	76,03
511955 (500MB)	1 sp	8	2,262958	1,691504	33,78	99,49
	10 sp	53	5,409291	2,541107	112,87	97,26
	100 sp	632	22,958459	8,336736	175,39	77,35
	1 lp	55	2,090559	0,968195	115,92	99,77
	10 lp	578	7,656948	2,818840	171,63	99,05
	100 lp	3914	15,797668	8,034843	96,61	76,22
1048906 (1GB)	1 sp	8	4,163488	2,823843	47,44	99,49
	10 sp	53	12,267914	5,024411	144,17	97,25
	100 sp	632	40,054606	15,947976	151,16	77,34
	1 lp	55	3,991538	1,955866	104,08	99,76
	10 lp	578	12,254003	4,495377	172,59	99,05
	100 lp	3914	32,940271	16,499932	99,64	76,22

(i) The *goto*, the *failure* and the *next-move* functions (executed during the pre-processing phase of the AC algorithm), which are all coded in C++ language;
(ii) The search function *AC_NextMove_search_AVX2* (the pattern-matching machine), which is vectorized by explicitly calling the processors SIMD instructions (AVX2) using low-level assembly code and available registers.

The three functions of the pre-processing phase should not pose portability issues since they are coded and compiled from a high-level language. The search function *AC_NextMove_search_AVX2*, as implemented, is supported by Intel®processors (such as Haswell, Broadwell, Skylake, Kaby Lake, Skylake-X, Coffee Lake, and Cannon Lake processors), AMD Excavator, Zen, and Zen+ processors, and later manufacturers' processor x86 microarchitectures (implementations of the x86 ISA) such as AMD Zen2, and Intel®Cascade Lake processors expected in 2019.

To overcome the portability issues, vector operations can be implemented by the compiler using automatic vectorization, after taking care of organizing data and loops in a convenient way. This is the most recommended method to employ vector instruction support, because cross platform porting is provided by the compiler.

7. Conclusion. In this paper, we have presented the vectorized Aho-Corasick algorithm in attempting to speed up the string-matching process. Experimental results clearly show that the vectorized version of the Aho-Corasick algorithm, using the *next-move* function, yields better performance in terms of speed whose running time was up to five (×5) of its original counterpart (not vectorized). The speed-up gained depends on the number of the patterns to search for and their length but not on the file size.

TABLE 5.2
Running times obtained while searching different file size for sets of 10 different short patterns

Text file size (KB)	Total patterns length (Bytes)	Number of States (MAXSTATES)	Time spent in state 0 (%)	T1(sec) AC NextMove (ASM)	T2(sec) AC NextMove (AVX2)	Speed-up(%) [AVX2/ ASM]	Speed-up(%) [AVX2/ ASM] TheoV
	48	39	98,23	0,073203	0,044321	65,17	484,49
	50	38	98,42	0,046869	0,029719	57,71	302,42
10	53	45	98,33	0,082621	0,048733	69,54	410,67
MB	53	50	97,02	0,098311	0,047394	107,43	570,69
	65	51	96,44	0,081480	0,048452	68,17	472,95
	82	73	93,83	0,118258	0,085747	37,92	454,41
	48	39	98,29	0,085187	0,046276	84,08	484,79
	50	38	98,55	0,631662	0,412058	53,29	302,62
100	53	45	98,46	0,836997	0,475502	76,02	411,45
MB	53	50	97,25	1,362945	0,580876	134,64	572,15
	65	51	96,41	0,961730	0,558828	72,10	472,60
	82	73	93,44	1,574822	1,024999	53,64	452,15
	48	39	98,30	11,628367	4,678585	148,54	484,93
	50	38	98,56	5,675574	3,606079	57,39	302,65
1	53	45	98,46	9,870739	5,026897	96,36	411,53
GB	53	50	97,26	12,450564	5,123501	143,01	572,14
	65	51	96,47	11,870859	5,670317	109,35	472,86
	82	73	93,43	14,372971	9,443510	52,20	452,22

TABLE 5.3
Running times obtained while searching different file size for sets of 10 different long patterns

Text file size (KB)	Total patterns length (Bytes)	Number of States (MAXSTATES)	Time spent in state 0 (%)	T1(sec) AC NextMove (ASM)	T2(sec) AC NextMove (AVX2)	Speed-up(%) [AVX2/ ASM]	Speed-up(%) [AVX2/ ASM] TheoV
	560	558	98,72	0,204947	0,061213	234,81	884,82
	466	465	98,90	0,186187	0,048387	284,79	886,51
10	409	402	95,22	0,134060	0,052976	153,06	767,76
MB	565	557	99,32	0,160171	0,028548	461,06	693,44
	373	362	94,65	0,078206	0,044593	75,38	577,11
	327	324	91,73	0,197331	0,099463	98,40	694,54
	560	558	98,97	1,965827	0,623113	215,48	887,10
	466	465	99,20	2,178835	0,612167	255,92	889,77
100	409	402	94,77	2,002379	0,649050	208,51	765,40
MB	565	557	99,54	1,389077	0,344041	303,75	695,20
	373	362	94,24	1,107682	0,562100	97,06	575,62
	327	324	91,17	1,847791	1,027445	79,84	689,78
	560	558	99,02	18,291795	5,667745	222,73	887,74
	466	465	99,26	20,000147	5,505017	263,31	890,33
1	409	402	94,79	17,471766	6,047175	188,92	765,57
GB	565	557	99,59	14,13522	2,825973	400,19	695,48
	373	362	94,25	13,016356	5,959448	118,42	575,69
	327	324	91,26	16,198667	9,344094	73,36	690,75

The use of vectorization results on a higher return in performance and efficiency that depend on the code structure. But, once again, the programmer must examine at first the structure of the code to vectorize in depth to identify parts (code fragments) that offer vectorization opportunities, and make the data structures in the code nearly fit the structure built into the hardware.

TABLE 5.4
Comparison of related work

Optimized implementations of the AC algorithm	Speed-up of Aho-Corasick Pattern Matching Machines by Rearranging States [6]	String matching with multicore CPUs: Performing better with the Aho-Corasick algorithm [11]	SIMD implementation of the Aho-Corasick algorithm using Intel AVX2 (Our implementation)
Experiment results	The elapsed time of string pattern-matching is reduced to 71(%) (Speedup 142(%)) in case of a random text and 55(%) (Speedup 110(%)) in case of a compressed text	Throughput ratio for 10MB pattern set (DNA sequences – short patterns) for 70 KB input size is 4.62 and for 30MB pattern set is 12.25. The throughput ratio is increasing with the increasing pattern size *(Throughput = Number of bytes in the input/ Total time for processing).*	Speedup between 30(%) and 195(%) while searching different text file size for sets of different patterns. Speedup between 37(%) and 148(%) while searching different text file size for sets of different short patterns. Speedup between 73(%) and 461(%) while searching different text file size for sets of different long patterns.

REFERENCES

[1] CORMAN, T. H., LEISERSON, C. E., RIVEST, R. L., & STEIN, C. (2002). Introduction to Algorithms-String matching. *EEE Edition, 2nd Edition, Page,* (906-907).

[2] BINSTOCK, A., & REX, J. (1995). Searching. *Practical Algorithms for Programmers* (pp. 95-171). Addison-Wesley Longman Publishing Co., Inc..

[3] AHO, A. V., & CORASICK, M. J. (1975). Efficient string-matching: an aid to bibliographic search. *Communications of the ACM*, 18(6), 333-340.

[4] WU, S., & MANBER, U.(1994). *A fast algorithm for multi-pattern searching.* (pp. 1-11). University of Arizona. Department of Computer Science.

[5] SALMELA, L., TARHIO, J., & KYTJOKI, J. (2006). Multi pattern string-matching with q-grams. *In the proc. of Journal of Experimental Algorithmic (JEA)*, 11, 1-19.

[6] COMMENTZ-WALTER, B. (1979, JULY). A string matching algorithm fast on the average. In *Proceeding of the 6th International Colloquium on Automata, Languages, and Programming* (pp. 118-132). Springer, Berlin, Heidelberg.

[7] AHO, A. V. (1990). Algorithms for finding patterns in strngs, Handbook of Theoretical Computer Science Vol A. *A, ed. J. van Leeuwen, ElsevierSciencePublishersB, 1990,* 257-297.

[8] JONY, A. I. (2014). Analysis of multiple string pattern matching algorithms. *International Journal of Advanced Computer Science and Information Technology (IJACSIT)*, 3(4), 344-353.

[9] NISHIMURA, T., FUKAMACHI, S., & SHINOHARA, T. (2001, NOVEMBER). Speed-up of Aho-Corasick pattern matching machines by rearranging states. In *Proceedings Eighth Symposium on String Processing and Information Retrieval* (pp. 175-185). IEEE.

[10] LIN, C. H., TSAI, S. Y., LIU, C. H., CHANG, S. C., & SHYU, J. M. (2010, DECEMBER). Accelerating string-matching using multi-threaded algorithm on GPU. In *Proceedings of IEEE Global Telecommunications Conference GLOBECOM 2010* (pp. 1-5). IEEE.

[11] ARUDCHUTHA, S., NISHANTHY, T., & RAGEL, R. G. (2013, DECEMBER). String matching with multicore CPUs: Performing better with the Aho-Corasick algorithm. In *2013 IEEE 8th International Conference on Industrial and Information Systems* (pp. 231-236). IEEE.

[12] INTEL (2016). An introduction to Vectorization with the Intel C++ Compiler. `https://software.intel.com/sites/default/files/An\_Introduction\_to\_Vectorization\_with\_Intel\_C++\_Compiler\_021712.pdf`. [Online; accessed 02 February 2016].

[13] GTZFRIED, J. (2012). Advanced Vector Extensions to accelerate crypto primitives. *Friedrich-Alexander-Universitt, Erlangen-Nrnberg. Bachelor Thesis.*

[14] VLADIMIROV, A., ASAI, R., & KARPUSENKO, V. (2015). ExpressingParallelism. *Parallel Programming and Optimization with Intel®XEON PHI™Coprocessors* (2nd ed., pp. 157-158). Colfax International.

[15] OXFORD E-RESEARCH CENTRE (2016). AVX Vectorisation and Cilk Plus. `http://www.oerc.ox.ac.uk/projects/asearch/software/avx`. [Online; accessed 26 April 2016].

[16] PAGE, D. (2009). Advanced Processor Design. *A Practical Introduction to Computer Architecture* (pp. 389-390). Springer Science & Business Media.

[17] JEFFERS, J., REINDERS, J., & SODANI, A. (2016). Introduction. *Intel®Xeon Phit Coprocessor High Performance Programming. Knights Landing Edition.* (pp. 11-12). Morgan Kaufmann.

[18] INTEL (2016). Intel®Advanced Vector Extensions (Intel®AVX).`https://software.intel.com/en-us/isa-extensions/intel-avx`. [Online; accessed 07 January 2016].

[19] KUSSWURM, D. (2014). X86-32 Core Architecture. *Modern X86 Assembly Language Programming: 32-bit, 64-bit, SSE, and AVX.* (pp. 01-03). Apress.

[20] Intel (2016). Intel®Architecture Instruction Set Extensions Programming Reference. https://www.naic.edu/\~\phil/software/intel/319433-014.pdf. [Online; accessed 06 January 2016].
[21] Knuth, D. E., Morris, Jr, J. H., & Pratt, V. R. (1977). Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2), 323-350.

Edited by: Dana Petcu
Received: June 6, 2019
Accepted: September 1, 2019

DOI 10.12694/scpe.v20i3.1574
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 3, pp. 577–590. http://www.scpe.org

AMIGM: ANIMAL MIGRATION INSPIRED GROUP MOBILITY MODEL FOR MOBILE AD HOC NETWORKS

JYOTSNA VERMA AND NISHTHA KESSWANI*

Abstract. The most widespread notion of mobility model is the representation of mobile nodes movement pattern in the wireless ad hoc networks which has a significant impact on the performance of the network protocols. In this paper, we have proposed an Animal Migration Inspired Group Mobility (AMIGM) model for mobile ad hoc networks based on the migration behavior of animals like, insects, flock of birds, schools of fishes, reptiles, amphibians, etc. The propound model tries to overcome the limitations of the existing mobility models, such as temporal dependencies, spatial dependencies, geographical restrictions and migration of nodes between the group of nodes so that it can realistically model the real world application scenarios. The proposed AMIGM model is based on Animal Migration Optimization algorithm, in which each group of nodes has two phases namely, Migration phase and Population updating phase. In the first phase, the model simulates the movement of nodes in the group from one position to another by obeying the swarming laws. In the second phase, the model simulates joining and leaving of the nodes in the group during migration. The protocol dependent and independent performance metrics of the proposed model are compared with Random Waypoint Mobility model and Reference Point Group Mobility model through ns-2 simulator.

Key words: Animal migration optimization, Mobility model, Ad hoc Networks, RWP, RPGM.

AMS subject classifications. 68M10

1. Introduction.

The evolution of Mobile Ad hoc Networks (MANETs) over the years is potentially one of the major advances in the history of wireless networking. MANET is basically an autonomous system of mobile nodes connected by wireless links without any fixed infrastructure. It is characterized by multi-hop wireless links, shared radio channel, distributed routing, packet switched network, quick and low cost of deployment, self-organization and maintenance [1]. These characteristics of MANETs led to a diverse range of applications, such as emergency and rescue operations, disaster recovery, transportation systems, military applications, security operations, environmental monitoring, conferences and smart homes. Nodes in MANETs are highly mobile and can move in any direction with any speed which leads to frequent path breaks and affects the performance of wireless ad hoc networks, such as connectivity of the nodes, communication traffic, network protocols, etc., [2, 3]. Mobility model is a critical and important parameter for a system that specifies the movement pattern of mobile nodes. Thus, it is significant to emulate the real world applications. Various individual as well as group mobility models exist in the literature that tries to realistically model the different movement patterns of the nodes, such as Random Waypoint (RWP) model [4] and Reference Point Group Mobility (RPGM) model [5] etc. The performance of wireless ad hoc networks varies considerably with different mobility models and their parameters during simulation. Thus, an efficient mobility model is required that can model the realistic scenarios.

In view of the necessity of developing efficient, realistic mobility model, the paper proposes an AMIGM model based on swarming rules and Animal Migration Optimization (AMO) algorithm [6] which is a new heuristic optimization method based on the animal swarm migration behavior. In the proposed model, each group of nodes in the network has two phases: Migration phase and Population updating phase. In the first phase, the model simulates the movement of nodes in the group from one position to another by obeying the swarming rules such as cohesion, homing, dispersion, collision avoidance, etc., to direct their movement in the network. In the second phase, the model simulates joining and leaving of the nodes in the group during migration. Thus, our model is capable of modeling the migration of nodes between the groups based on AMO algorithm and can avoid inter-intra group collision, environmental obstacles which mathematically model the movement pattern of the nodes in mobile ad hoc networks.

1.1. Motivation.

Mobility is the salient feature of MANETs. So, researchers have designed many mobility models over the past years to mimic the realistic scenarios, but fail to recreate the realistic applications for the MANETs, such as migration of nodes from one group to another group, collision avoidance among the nodes,

*Department of Computer Science, Central University of Rajasthan, Ajmer, Rajasthan, India. (jyotsna.verma.mca@gmail.com).
This research is supported by the NFO Fellowship under the University Grants Commission

avoidance of geographical restrictions etc. Hence, to realistically represent the movement pattern of the nodes, researchers are eventually approaching towards the bio inspired techniques where the solutions of the problems are inspired from swarm behavior of natural systems, like insects, flock of birds, schools of fishes, reptiles, amphibians, etc. [7]. Characteristics of bio inspired approaches, like cooperative movement, no permanent membership, speed, adaptation, scalability, fault tolerance, modularity, parallelism and autonomy etc., suit the application of ad hoc wireless networks.

There are many mobility models [8, 9, 10, 11] in the literature that use nature inspired techniques to model the realistic scenarios and till date, to the best of our knowledge, there is no generalized mobility model for all the application scenarios. The approach of our proposed mobility model is based on the animal migration behavior ecology which can be found in all animal groups, like fish, reptiles, amphibians, insects, etc. Generally, the cause of animal migration is due to climate change, food, seasons, etc. So, the main attribute of the work is to model the migration behavior of the animals from one group to another group to achieve realistic mobility model.

1.2. Contributions. The principal contributions of this paper are as follows:

1. We have proposed an Animal Migration Inspired Group Mobility (AMIGM) model for MANETs based on the AMO approach and some swarming laws for the governing of the group mobility. It is capable of modeling the migration behavior of the nodes from one group to another group by replacing the worst fitness nodes of one group with the best fitness nodes from the other group of nodes.
2. Mathematical models of the group formation and migrating procedures are presented.
3. We have compared AMIGM model with the two prevailing existing mobility models: RWP [4] and RPGM [5].
4. Finally, the effect of mobility models on network performance with respect to the mobility under Dynamic Source Routing (DSR) protocol is evaluated.
5. The simulation results for the proposed AMIGM model shows the least average energy consumption with highest average link duration and average degree of spatial dependence when compared with existing models.

The rest of the paper is structured as follows: Section 2 highlights the related works. Section 3 presents the proposed Animal Migration Inspired Group Mobility (AMIGM) model. Section 4 presents the algorithm for the proposed AMIGM model and Section 5 analyses the time complexity of the proposed AMIGM algorithm. Section 6 presents the simulation environment and results of the protocol dependent and independent performance of mobility models in MANETs under the DSR routing protocol. Section 7 presents the movement pattern of the nodes correspond to the different mobility models and Section 8 summarizes the paper.

2. Related Work. The emergence of the phenomenon commonly known as mobility model represents the movement pattern of the nodes and how their velocity, acceleration, and location changes over time. Considering the mobility characteristics of the nodes in MANETs, formulation of the realistic mobility model is very important as an unrealistic model gives unrealistic results which will ultimately affect the performance of the mobility models. There are many mobility models [3, 4, 5, 12, 13] in the literature that try to create the realistic application scenarios.

Broadly, mobility models are classified into three categories: Trace based mobility models, Stochastic/Individual mobility models and Synthetic/Group mobility models. Trace based mobility models are based on real traces, but due to the limited public repository of traces, it is difficult to model the nodes in real world scenarios. Stochastic mobility models are idealistic models where nodes move independently within the network without imposing the constraints like obstacles, pathways, etc. Random walk mobility model [14], Random waypoint mobility model [4], Random direction mobility model [15] etc., are some of the stochastic/individual/random based mobility models. The Random walk mobility model also called as Brownian motion is the most popular and simple mobility model which is first described mathematically by Einstein in (1926). In this, mobile nodes independently move from one location to another location with randomly chosen direction and speed from predefined ranges $[0, 2\pi]$ and $[minspeed, maxspeed]$ respectively. Another popular mobility model is the Random waypoint mobility model in which nodes independently move around the simulation area, including pause time between randomly chosen direction and speed. The Node begins its movement by staying at one location for specific pause time say for t second and then it moves again to another location within the simulation bound-

ary in an inconstantly chosen direction and speed from the ranges between $[0, 2\pi]$ and [minspeed,maxspeed] respectively. The Random direction mobility model, on the other hand, chooses a random direction and move along that direction until it reaches the simulation boundary. On reaching the simulation boundary, it again chooses a direction and repeats this process until the simulation time is reached. Random movement of nodes sometimes creates the interference in the network. Various mobility models found in the literature deals with the interference in the network. In [16] authors map the variation of distance caused by the nodes mobility to the variation in channel gain for describing the distinctive nature of the interference statistics of random networks. For handling complex mobility and predicting time varying interference authors in [17] proposes a general mobility model for a finite number of nodes using a general-order linear model in MANETs.

The individual mobility models are simple to implement, but they do not represent the group mobility behavior of the nodes in the network. This induces the development of stochastic/group mobility models. Group mobility models are the mathematical models which are spatially dependent i.e., they capture the group mobility behavior of the nodes, such as Exponential Correlated Random Mobility (ECRM) model [5], Pursue mobility model [18, 19] Column mobility model[18, 19], Reference point group mobility model [5] Reference Velocity Group Mobility model [20] and others. In Exponential correlated random mobility model, movement of the mobile nodes is directed by complex motion function, whereas the nodes in the Pursue mobility model, moves with little randomness towards the particular target. Nodes in the Column mobility model move in a forward direction around the line (or column). One of the most popular group mobility model is the Reference point group mobility model. In this, each node in the group is randomly distributed around the predefined reference point and follows the logical center of the group which decides the groups motion behavior along with a speed and direction via group motion vector $\overrightarrow{\mathrm{GM}}$. Reference Velocity Group Mobility model is another stochastic mobility model that uses the group velocity vector at time t to represent the group motion behavior.

Apart from individual and group mobility models, there are various other kinds of mobility models such as Obstacle mobility models like Pathway, Manhattan, Freeway mobility model, etc., which can model real world scenarios by avoiding environmental obstacles [21]. Map based mobility models are based on the traffic and user mobility patterns in order to evaluate and deal with networking issues with the wireless networks. A disaster aware mobility model is designed for flying Ad hoc Networks which is is emulated using map oriented navigation of nodes with realistic disaster situation [22]. In Social network based mobility model follows social network theories and very effectively captures the behavior of movement based on human decisions and socialization behavior like disaster relief, battlefield, etc. In [23] author proposed an N-body mobility model in which traces of a small number of nodes were captured for forming group behavior in order to reproduce them in the large population mobility traces. Inherently, social network theory studies the mobility pattern of the nodes based on social behavior for analyzing and formulating the network protocols in ad hoc networks.

However, our proposed AMIGM model is capable of modeling the migration of nodes from one group to another by replacing the worst fitness nodes and avoids the geographical restrictions, inter and intra group collision which was not present in the above existing mobility models. Moreover, AMIGM, RWP and RPGM models are evaluated with the protocol dependent and independent performance metrics under the DSR routing protocol.

3. Animal Migration Inspired Group Mobility Model. This Section discusses the proposed Animal Migration Inspired Group Mobility (AMIGM) model which is based on Animal Migration Optimization (AMO) algorithm [6] and some swarming laws for the governing of the group mobility.

The AMIGM model has two phases (1) Migration phase and (2) Population updating phase and it begins with the initialization process in which nodes are initially randomly distributed throughout the simulation area, according to a uniform distribution. In the migration phase, nodes within the group should follow three basic swarming rules: (1) they should remain close to their neighbors, (2) the whole group should move in the same direction to a common destination and (3) they should avoid inter and intra group collision. For the first and second rules, nodes should be kept within the sensor range ρ) of each other by calculating the distance of the nodes from each other through the Euclidean distance given below:

$$d_{min} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} \leq \rho \tag{3.1}$$

Each group of nodes, then decides to move with the constant speed to a common destination or signal

source which is randomly distributed throughout the simulation area with the desired homing heading angle ψ_{Hi}, given in equation (3), mathematically proposed by Sharma and Ghose [24]].

The nodes in the network will be in the communication range of each other and will remain in one group (G) if the distance d_{min} between the nodes is less than or equal to the sensor range ρ. After the formation of the group, neighbor list is defined for each node within the group (G). Once the construction of the neighborhood topology gets completed, each node within the group moves from one position to another according to the neighbors position within the simulation area following the basic swarming laws. The i^{th} node position of the group after $t + \Delta t$ time is updated according to the equation (2) and the updated position of the i^{th} node will be:

$$X_{(i,t+\Delta t)} = X_{(i,t)} - \delta.(X_{(neighborhood,t)} - X_{(i,t)}) \leq \rho \tag{3.2}$$

where, $X_{(neighborhood,t)}$ is the current position of the neighborhood node and δ is the random number generator using Gaussian distribution.

Each group of nodes, then decides to move with the constant speed to a common destination or signal source which is randomly distributed throughout the simulation area with the desired homing heading angle ψ_{Hi}, given in equation (3), mathematically proposed by Sharma and Ghose [24].

$$\psi_{Hi} = arctan((Y_{Si} - Y_i)/(X_{Si} - X_i)) \tag{3.3}$$

where, (X_{Si}, Y_{Si}) are the coordinates of the signal source.

For the third rule, nodes within the group avoids the collision with each other by maintaining the minimum required distance d_{min} between the nodes with the desired dispersion heading angle ψ_{Di}, given in equation (4) [24].

$$\psi_{Di} = arctan((Y_i - Y_{Ci})/(X_{Ci} - X_i)) \tag{3.4}$$

where, (X_{Ci}, Y_{Ci}) are the centroid of all the nodes within the group.

To avoid the collision between the groups they have to maintain the safe distance with the other group along with the minimization of group deviation from its current direction. The group first determines whether the other group is within the collision avoidance range ρ_{col} or not. To maintain the brevity of the paper detailed explanation about this rule is not mentioned here, but can be found in [24]. For the collision avoidance rule, avoidance rule weight W_{AVi} is given by:

$$W_{AVi} = \begin{cases} 1 & \text{if } P_i - C_{\rho,i} \leq \rho_{col} \\ 0 & \text{if } P_i - C_{\rho,i} > \rho_{col} \end{cases} \tag{3.5}$$

The avoidance rule weight W_{AVi} is used to calculate the desired heading angle ψ_{AVi} :

$$\psi_{AVi} = f_1(\frac{n}{2}) + \psi_i \tag{3.6}$$

where, ψ_i heading angle of the i^{th} node and f_1 is the direction of the turn in the horizontal plane and is given by:

$$f_1 = \begin{cases} -1 & \text{if } \frac{\pi}{4} < \psi_i \leq \frac{3\pi}{4} and \text{ if } x_i \geq X_{ca} \\ +1 & \text{if } \frac{\pi}{4} \leq \psi_i \leq \frac{3\pi}{4} and \text{ if } x_i < X_{ca} \\ +1 & \text{if } \frac{5\pi}{4} < \psi_i \leq \frac{7\pi}{4} and \text{ if } x_i \geq X_{ca} \\ -1 & \text{if } \frac{5\pi}{4} < \psi_i < \frac{7\pi}{4} and \text{ if } x_i < X_{ca} \\ -1 & \text{if } \frac{3\pi}{4} < \psi_i \leq \frac{5\pi}{4} and \text{ if } y_i \geq Y_{ca} \\ +1 & \text{if } \frac{3\pi}{4} \leq \psi_i < \frac{5\pi}{4} and \text{ if } y_i < Y_{ca} \\ +1 & \text{if } \frac{7\pi}{4} < \psi_i \leq \frac{\pi}{4} and \text{ if } y_i \geq Y_{ca} \\ -1 & \text{if } \frac{3\pi}{4} \leq \psi_i < \frac{5\pi}{4} and \text{ if } y_i < Y_{ca} \end{cases} \tag{3.7}$$

In addition to the basic swarming rules, there is requirement of avoiding environmental obstacles for modeling the real world scenarios. In the proposed mobility model, the nodes avoid the collision with the environmental obstacles. The obstacles are fixed in our model and the nodes will avoid the obstacles if it detects an obstacle within its sensor range ρ; same as they avoid group collision mentioned in [24]. The desired obstacle avoidance rule heading angle ψ_{OBi} is determined by [9]:

$$\psi_{OBi} = w_{ob}(\frac{n}{4}) + \psi_i \tag{3.8}$$

where, w_{ob} is obstacle avoidance rule weight.

Superposition is used to calculate the desired heading angle ψ_{req} given below as there are more than one swarming requirement rules in the proposed AMIGM model.

$$\Delta\psi_{req} = w_1(\psi_{req1} - \psi) + w_2(\psi_{req2} - \psi) + ... + w_n(\psi_{reqn} - \psi) \tag{3.9}$$

The rule weights $w_1, w_2, ., w_n$ are constants and varied to obtain different basic behaviors, where $|w_i| \leq 1$ and ψ is the heading angle of the node.

In the population updating phase, some nodes will leave the group and some nodes will migrate from one group to another group. The nodes with best fitness value will replace the nodes with worst fitness value during the migration of the nodes. In the AMIGM model, each node of the group randomly moves within the simulation area and if the groups of node is within the specified range ρ_G, the nodes will migrate to another group and replace the node with worst fitness value with the probability p_a.

$$D_{min} = \sqrt{(C_{Gxi} - C_{Gy_i})^2} \leq \rho_G \tag{3.10}$$

where, (C_{Gxi}, C_{Gy_i}) is the centroid of the group of nodes and ρ_G is the sensor range of the group of nodes.

The residual energy of the node is used as the fitness function for evaluating the solution. The AMIGM model makes use of nodes energy as the fitness function for evaluating the solution. Suppose N nodes $N_1, N_2, , N_n$ of group G_i are randomly distributed in the simulation area and each node of the group G_i has initial energy, E_0. The random m destination or target $d_1, d_2, , d_m$ is defined for each group of nodes G_i for visiting within the simulation environment. The visiting matrix $V_{i,j}$ of the group of nodes G_i is given in equation (11). If the nodes in the group G_i reach or visit the randomly selected destination d_j the visiting matrix $V_{i,j}$ will be:

$$V_{i,j} = \begin{cases} 1 & \text{if } N_i of each G_i visits the target d_j \\ 0 & otherwise \end{cases} \tag{3.11}$$

where $i = 1, 2, , n$ and $j = 1, 2, , m$

If b_i is the initial power of the battery and e_i is the energy consumption rate then the energy of the node will be:

$$b_i' = \frac{b_i}{e_i} \tag{3.12}$$

Hence, for the evaluation of the solution the energy of the node will be the fitness function which is given by:

$$f_x = (V_{i,j} * b_i') \tag{3.13}$$

The probability of a node being selected in the BIGM model with f_x as the fitness of a node is:

$$P_a = \frac{f_x}{\sum_{i=1}^{n} f_y} \tag{3.14}$$

For the worst fitness, the probability P_a is 1/N where, N is the number of nodes in the network and for the best fitness the N probability P_a is 1.

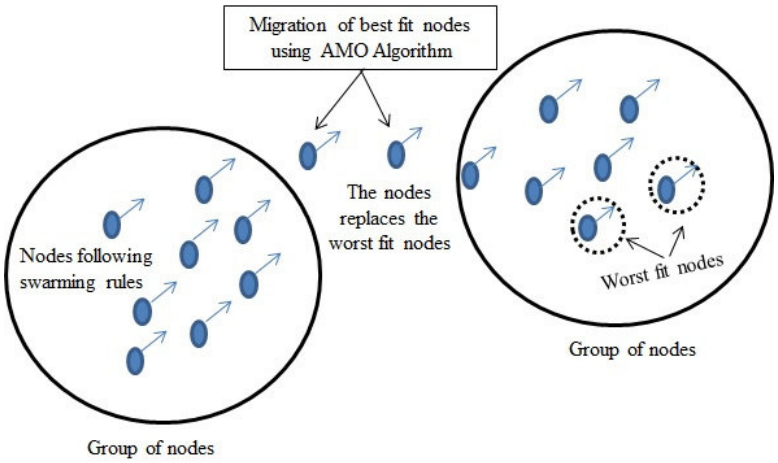


FIG. 3.1. *Migration of the nodes using Animal Migration Optimization Algorithm.*

The proposed AMIGM algorithm will select the best fitness value nodes i.e., the nodes with maximum amount of energy to migrate to another group of nodes. If the probability P_a of a node is greater than that of a randomly generated number (i.e., between 0.0 and 1.0) then that node will be selected for the migration with best fitness value. The selected node is then compared with the worst fitness node from the other group, and if the selected node has better fitness value, then the selected node will migrate into that group replacing the worst fitness node otherwise selected node will not migrate. After migrating to other group the selected node move towards the target or destination d_j, along with that group.

AMIGM model is designed for modelling the movement pattern of the nodes in wireless ad hoc networks and could be applied to a diverse range of the mobile ad hoc application scenarios. If we take a scenario of military application, where there are different teams like a disaster management team, rescue team, medical assistant team etc., working on the same battlefield, then there are chances that the medical assistant team may give some assistance to the disaster recovery team depending upon the application scenario. Then, some or whole medical assistant team may join the rescue team over the same disaster recovery area and work with them as a whole by replacing the worst node (in terms of energy of the nodes) of the rescue team by the best node of the medical assistant team. As there are the chances that either of the teams may have nodes with less energy while moving within the application scenario, then the nodes with the best energy will replace the worst energy nodes from either of the teams as required. It can be seen that in these types of application scenarios AMIGM model can work well.

4. Animal Migration Inspired Group Mobility Model Algorithm. The Animal Migration Inspired Group Mobility Model (AMIGM) is discussed above in Section 3 and the algorithm for the above proposed model is as follows:

Algorithm 1 Animal Migration Inspired Group Mobility Model for Mobile Ad hoc Networks

Input: Number of nodes N, Number of group of nodes G and Sensor range ρ.
Output: Migration of nodes to another group by replacing the worst fit nodes.

1: **begin**
2: Create a network of nodes N and set initial position of nodes according to a uniform distribution.
3: Define a neighbor list for each node.
4: Define group G for the network by calculating distance and comparing it with sensor range ρ of each node N in the network.
5: Randomly select a destination for the group.
6: Allot energy to each node N in the group.
7: **while** maximum simulation time **do**

```
8:        for j=1 to G do
9:            for i=1 to N do
10:               Update Position();
11:               Homing requirement();
12:               Dispersion();
13:               Collision Avoidance();
14:               Obstacle avoidance();
15:               Sort all the nodes according to their fitness.
16:               if (rand P_a)and(D_min ≤ ρ) then
17:                   Evaluate the fitness.
18:                   if X_(i,j+1) has better fitness than the X_(i,j) then
19:                       Select X_(i,j+1) as best node and replace it with X_(i,j) ; the worst node of the group.
20:                   else
21:                       No migration will take place.
22:                   end if
23:               end if
24:           end for
25:       end for
26:   end while
27: end
```

5. Complexity Analysis. This section analyses the time complexity of the proposed AMIGM algorithm. In the proposed algorithm for AMIGM model (see algorithm 1), neighbors of the node will get updated each time it iterates through n nodes; hence, the complexity will be $O(n^2)$ for the updated list of neighborhood. The selection of the best fit nodes from the group (G_i) and comparing that best fit node with the worst fit node, the algorithm requires the complexity of $O(n^2)$ in worst case and to replace the worst fit node with the best fit selected nodes constant time will be required during migration of nodes in the algorithm. Hence, for each group (G_i) the migration algorithm iterates through n nodes and has the complexity of $O(n^2)$. Therefore, the complexity of the proposed AMIGM algorithm will be $O(n^2)$.

6. Simulations and Results. This Section presents the simulation environment and configurations used for testing the AMIGM model. Also, a comparative result of the AMIGM model with the RWP and RPGM model is evaluated under the DSR routing protocol.

6.1. Simulation Environment. The simulator used for modeling the proposed mobility model was ns-2 [25] and cbrgen tool of ns-2 was used for generating the communication traffic. We have randomly chooses the source and destination pairs with 30 independent cbr traffic sources and overall 30 connections. Each simulation result of the mobility models used setdest tool to generate mobility scenarios and has eight different mobility scenario files for the maximum speed of 5, 10, 15, 20, 25, 30, 35 and 40 m/s with a 0 s pause time. Also, we have considered 10 randomly distributed obstacles within the simulation area in our proposed model. The performance metrics of the proposed model along with the RWP and RPGM model under the DSR routing protocol are compared. The nominal values of the weights discussed in Section 3 are assumed to be $w_c = 0.5$, $w_h = 0.7$, w_d =0.5 and $w_{av} = 0$, without collision avoidance scheme. With collision avoidance scheme values will be $w_c = 0.25$, w_h=0.35, w_d=0.5 and $w_a v$=1. The values of the weights are borrowed from Sharma and Ghose [24]. For the obstacle avoidance scheme if an obstacle is detected within the sensor range of the nodes the value of w_{ob} will be 1 and if no obstacle is detected then the value of w_{ob} will be 0. The parameters used for conducting the simulation of mobility models are given in Table 6.1 and the distribution of the nodes in the mobility models are given in Table 6.2.

6.1.1. Average Degree of Spatial Dependence. It is the measure of similarities of node velocities of the two neighboring nodes. In AMIGM model, nodes are spatially correlated as the group of nodes try to keep up common heading angle. As can be seen in Fig. 6.1 AMIGM mobility model has the highest average degree of spatial dependence compared to RPGM and RWP as in the AMIGM model all the nodes belonging to a group move towards the same direction with the same speed. For instance, AMIGM shows the highest average degree

TABLE 6.1
Simulation Parameters.

Simulation Parameters	Specifications
Simulation Time	800 s
Field Dimensions	(1500m, 1500m)
Wireless Antenna	Omni directional antenna
Network Interface	Wireless
Channel	Wireless
Initial Energy	60 Joules
Transmission Range	250 m
Interface Queue	CMU Priority queue, Queue/DropTail/PriQueue
Interface queue length	50
Propagation path loss	Two ray ground
MAC Protocol	802.11
Routing Protocol	DSR, AODV, DSDV
Data Rate	200 kb
CBR Packet Size	64 bytes
UDP Packet Size	512

TABLE 6.2
Node Distribution in Groups with Different Node Density.

Mobility Models	Groups	Nodes/Group
RWP	No Groups	60
RPGM	1	35
	2	25
AMIGM	1	35
	2	25

of spatial dependence of 0.51 at the maximum speed of 25 m/s as compared to RPGM and RWP which shows 0.28 and 0.00052 average degree of spatial dependence respectively.

6.1.2. Average Link Duration. Link duration is the measure of the link duration between the nodes in the network. Nodes in the ad hoc wireless networks frequently change the topology and hence, the nodes exhibit more links up/down with high mobility. But the nodes in the AMIGM mobility model remain close to each other because of the cohesive property explained in Section 3 and thus, AMIGM mobility model shows the highest average link duration because of the fewer links up/down between the nodes compared to RPGM and RWP mobility models (see Fig. 6.2). It is desirable that the average link duration remains high for the energy constraint MANETs. For instance, the average link duration of the AMIGM mobility model is 17.2 s at the maximum speed of 25 m/s as compared to RPGM and RWP mobility model which shows 10.34 s and 3.37 s respectively. As shown in Fig. 6.2 with high mobility the average link duration between the nodes decreases with an increase in maximum speed as they exhibit more links up/down.

6.1.3. Packet Delivery Ratio. The packet delivery ratio is defined as the ratio of the successfully received packets by the destination to the generated packets by the source. In Fig. 6.3 we can see that RPGM mobility model has the highest packet delivery ratio as compared to RWP and AMIGM models for DSR routing protocol as nodes in AMIGM model avoids the inter-group communication as a result of which most of the routing packets are dropped. But in some instances, AMIGM shows the highest packet delivery ratio than RPGM.

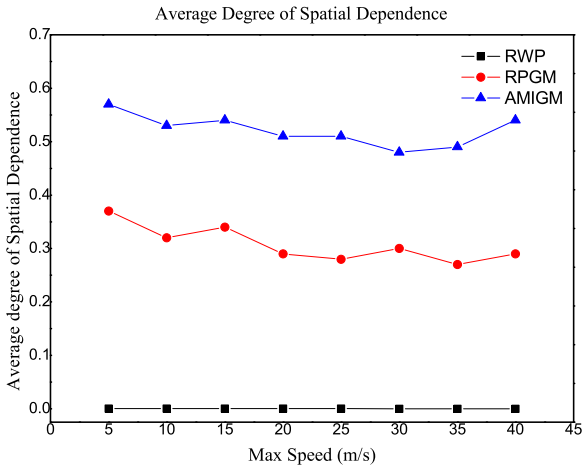


FIG. 6.1. *Average Degree of Spatial Dependence with the varying network load across various mobility models.*

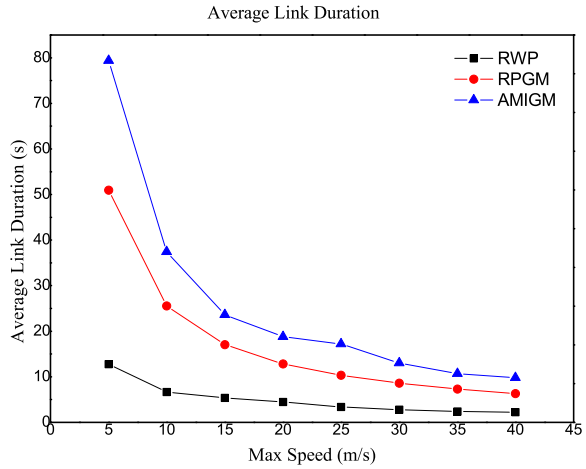


FIG. 6.2. *Average Link Duration with the varying network load across various mobility models.*

For instance, in Fig. 6.3 for the DSR routing protocol when the maximum speed is 5 m/s, AMIGM shows the highest packet delivery ratio of 1.11 compared to RWP and RPGM which shows 0.34 and 1.21 respectively. Thus, from the Fig. 6.3 it is clear that in some instances, AMIGM shows the highest packet delivery ratio than RWP and RPGM mobility as in those instances, due to the migration of nodes from one group to another group, the nodes remain within the transmission range of each other.

6.1.4. Normalized Routing Overhead . Normalized routing overhead is defined as the total number of routing packets sent per data packet to the destination. In Fig. 6.4 we can see that RWP mobility model has the highest normalized routing overhead as nodes move independently in RWP and thus route fluctuate more rapidly compared to RPGM and AMIGM models for the DSR routing protocol. For instance, in Fig. 6.4 for DSR routing protocol, when the maximum speed is 25 m/s the AMIGM shows the routing overhead of 0.75 kbps whereas RWP and RPGM show the routing overhead of 1.55 kbps and 0.59 kbps respectively. Thus, from

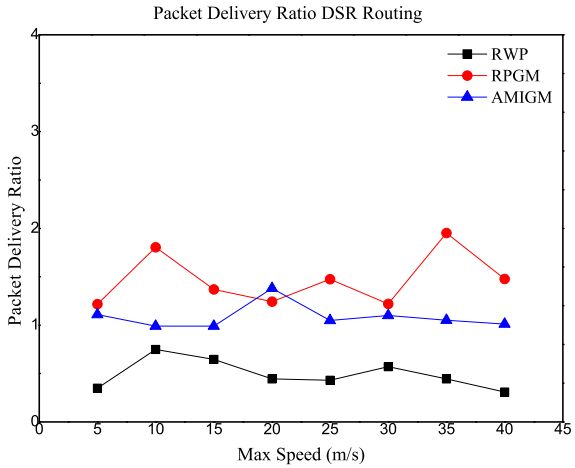


FIG. 6.3. *Packet Delivery Ratio (in fraction) across various mobility models with maximum speed for DSR with AMIGM.*

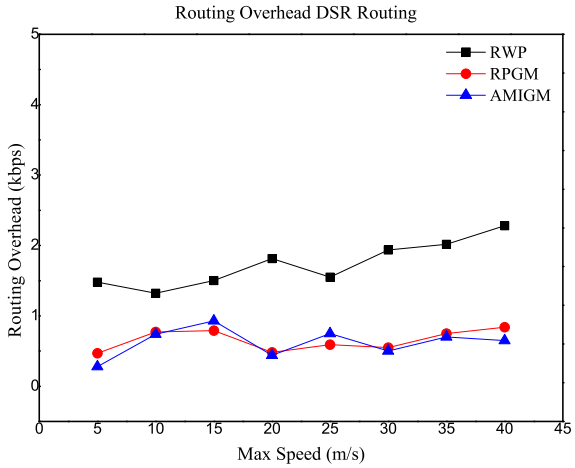


FIG. 6.4. *Normalized Routing Overhead (control packets per data packets sent) across various mobility models with maximum speed for DSR with AMIGM.*

the Fig. 6.4, it is clear that after RWP, AMIGM shows the highest normalized routing overhead as compared to RPGM model.

6.1.5. End-to-End Delay . End-to-End Delay is defined as the amount of time, a bit of data will take to travel between the two communicating nodes. In Fig. 6.5 we have observed that the RWP mobility model shows the maximum delay as compared to RPGM and AMIGM models for the DSR routing protocol. For instance, in Fig. 6 at the maximum speed of 40 m/s RWP, RPGM and AMIGM shows the delay of 0.80 s, 2.51 s, and 2.72 s respectively. Thus, from the Fig. 6.5, it is concluded that AMIGM shows the maximum delay compared to the RPGM model because with the inter-group communication the connection setup delay is high.

6.1.6. Throughput. Throughput is defined as the amount of data successfully sent from one node to another in a specified period of time. In Fig. 6.6, we have observed that the RPGM mobility model has the

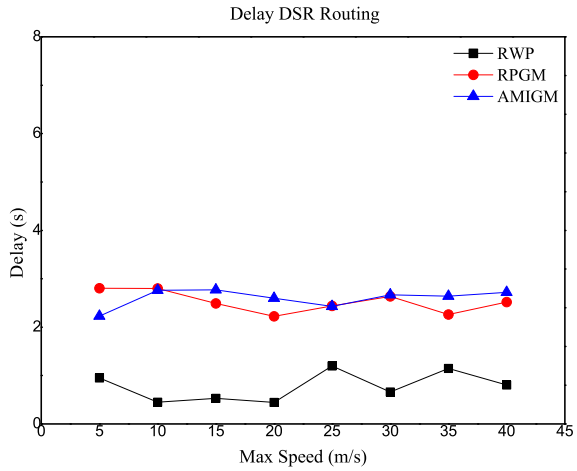


FIG. 6.5. *End-to-End Delay (in fraction) across various mobility models with maximum speed for DSR with AMIGM.*

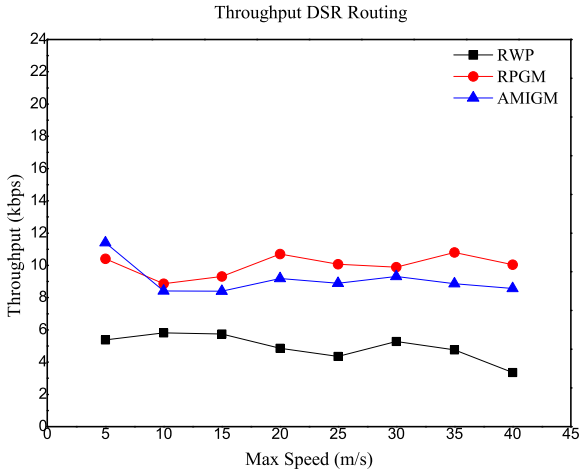


FIG. 6.6. *Throughput (kbps) across various mobility models with maximum speed for DSR with AMIGM.*

highest throughput as compared to RWP and AMIGM models for the DSR routing protocol because AMIGM model avoids the inter-group communication. In Fig. 6.6 with DSR routing protocol at the maximum speed of 5 m/s AMIGM shows the throughput of 11.4 kbps whereas RWP and RPGM models shows the throughput of 5.38 kbps and 10.4 kbps respectively. Thus, from the Fig. 6.6, it is concluded that in some instances, AMIGM shows the maximum throughput than RPGM model as the nodes in AMIGM model are within the transmission range of each other due to the migration of nodes from one group to other.

6.1.7. Average Energy Consumption. It is the amount of energy consumed during the movement of nodes in the network. In Fig. 6.7 we have observed that the RPGM mobility model has the highest energy consumption as compared to RWP and AMIGM models for the DSR routing protocol. In Fig. 6.7with DSR routing protocol at the maximum speed of 35 m/s AMIGM shows the average energy consumption of 57.62 joules whereas RWP and RPGM models show the average energy consumption of 59.00 joules and 58.12 joules

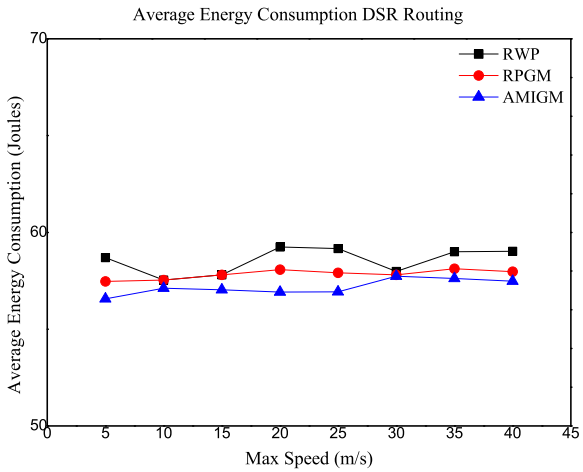


FIG. 6.7. *Average energy consumption (joules) across various mobility models with maximum speed for DSR with AMIGM.*

respectively. Thus, from the Fig. 6.7, it is concluded that AMIGM shows the minimum average energy consumption than RPGM model as the nodes in AMIGM model has the highest average spatial dependences and link duration. In addition, it also avoids the obstacle collision, inter and intra group collision between the nodes because of that least energy is consumed by the AMIGM model.

7. Movement Pattern of Nodes in Mobility Models. The mobile nodes in wireless ad hoc networks exhibit different movement pattern for modeling the realistic scenarios. Mobility models with their different parameters exhibit different characteristics which in turn affect the network protocols and can be used to evaluate network protocols for different application scenarios.

Following are some mobility model's characteristics for modelling the realistic scenarios:

1. Temporal dependencies: Mobility behavior in realistic scenarios, such as sudden change in acceleration, sharp turns and sudden stop may not occur frequently and the real movement of nodes defines the temporal dependencies based on the past movement of nodes. Random based mobility model are not able to model the real world scenarios as the velocity in these models has memoryless behavior.
2. Spatial dependencies: In most real world scenarios the nodes hardly move independently rather they move in a correlated fashion, such as military operations, emergency operations, conferences, museum touring and security operations. The mobility of the nodes depends on the mobility behavior of neighboring nodes which is absent in random based mobility models because the nodes move independently.
3. Geographical dependencies: Real world scenarios have many environmental restrictions like, buildings, trees, vehicles etc., and so, mobility models must take geographical restrictions into consideration while designing the models. Random mobility models do not take geographical restriction into consideration as they move freely within the simulation area and fail to model the real world scenarios.

The comparisons of different mobility models with the proposed AMIGM model under temporal, spatial dependencies and geographical restrictions are presented in Table 7.1.

As discussed above, random based mobility models are not spatial and temporal dependent, but nodes in synthetic mobility models like RPGM model move in a group and are correlated with each other without imposing geographical restrictions, hence RPGM model is spatial dependent. The proposed AMIGM model is temporal as well as the spatial dependent because nodes in AMIGM model move in a group as a whole by using basic swarming laws and movement of nodes do not include sharp turns rather they move incrementally while avoiding environmental obstacles present in the real world scenarios.

TABLE 7.1
Comparison of Mobility Models

Mobility Models	Temporal Dependency of Velocities	Spacial Dependency of Velocities	Geographical Restrictions
RWP	No	No	No
RPGM	No	Yes	No
AMIGM	Yes	Yes	Yes

8. Conclusion. A realistic mobility model is very significant for the evaluation of the performance of network protocols and validation of real world traces in the wireless ad hoc networks. This paper proposes an Animal Migration Inspired Group Mobility Model (AMIGM) which tries to mimic the movement of the real mobile node during migration from one group to another using AMO algorithm. The proposed model follows the simple swarming rules for the movement and the formation of the group of nodes. Also, it is capable of avoiding inter-intra group collision, environmental obstacles and can stimulate the migration of nodes between the groups by replacing the worst fitness nodes. We have simulated and compared the performance of the proposed AMIGM model with the RWP and RPGM mobility models under the DSR routing protocol and connectivity metrics. The simulation results for the proposed AMIGM model shows that the model has highest average link duration and average degree of spatial dependence with least average energy consumption when compared with other existing mobility models (RWP and RPGM) under the DSR routing protocol.

REFERENCES

[1] C. S. R. MURTHY, *Ad hoc wireless networks: Architectures and protocols*, Pearson Education, India, 2004.
[2] T. CAMP, J. BOLENG, V. DAVIES, *A survey of mobility models for ad hoc network research*,Wireless communications and mobile computing,2(5) (2002), pp. 483-502.
[3] M. L. SICHITIU, *Mobility Models for Ad hoc Networks*, in Guide to Wireless Ad Hoc Networks, Springer, London, 2009, pp. 237-147.
[4] D. B. JOHNSON, AND D. A. MALTZ, *Dynamic Source Routing in Ad hoc Wireless Networks*, in Mobile Computing, Springer, Boston, MA, 1996.
[5] X. HONG, M. GERLA, G. PEI, AND C. C. CHIANG, *A Group Mobility Model for Ad hoc Wireless Networks*, in Proceedings of the 2nd ACM International Workshop on Modeling, Analysis and Simulation of Wireless and mobile Systems, ACM, August 1999, pp. 53-60.
[6] X. LI, J. ZHANG, AND M. YIN, *Animal migration optimization: an optimization algorithm inspired by animal migration behavior*, Neural Computing and Applications, 24(7-8) (2014), pp.1867-1877.
[7] J. VERMA, AND N.KESSEANI, *A Review on Bio-Inspired Migration Optimization Techniques*, International Journal of Business Data Communications and Networking (IJBDCN), 11 (2015), pp. 24-35.
[8] B. ZHOU, K. XU, AND M. GERLA, *Group and Swarm Mobility Models for Ad hoc Network Scenarios using Virtual Tracks* in Military Communications Conference, MILCOM 2004, IEEE, 1 (October 2004), pp. 289-294.
[9] S. MISRA, AND P. AGARWAL, *Bio-Inspired Group Mobility Model for Mobile Ad hoc Networks based on Bird-Flocking Behavior*, Soft Computing, 16(3) (2012), pp. 437-450.
[10] J. HUO, B. DENG, S. WU, J. YUAN, AND I. YOU, *A topographic-Awareness and Situational-Perception based Mobility Model with Artificial Bee Colony Algorithm for Tactical MANET*, Computer Science and Information Systems, 10(2) (2013), pp. 725-746.
[11] J. VERMA, AND N. KESSWANI, *BIGM: A Biogeography Inspired Group Mobility Model for Mobile Ad Hoc Networks*, International Journal of Wireless Information Networks, 25(4) (2018), pp. 488-505.
[12] F. BAI, AND A. HELMY, *A Survey of Mobility Models*, in Wireless Adhoc Networks, University of Southern California, USA, 206, 147.
[13] J. M. NG AND Y. ZHANG, *Reference region group mobility model for ad hoc networks*, in Proceedings of Second IFIP International Conference on Wireless and Optical Communications Networks, IEEE, 2005, pp. 290-294.
[14] A. EINSTEIN, *Investigations on the Theory of the Brownian Movement*,Courier Corporation, 1956.
[15] E. M. ROYER, P. M. MELLIAR-SMITH, AND L. E. MOSER, *An Analysis of the Optimum Node Density for Ad hoc Mobile Networks*, in Communications, 2001, ICC 2001, IEEE International Conference on, 3 (2001), pp. 857-861.
[16] Z. GONG, AND M. HAENGGI, *Interference and outage in mobile random networks: Expectation, distribution, and correlation*, IEEE Transactions on Mobile Computing, 13(2) 2014, pp. 337-349.
[17] Y. CONG, X. ZHOU, AND R. A. KENNEDY, *Interference prediction in mobile ad hoc networks with a general mobility model*,IEEE Transactions on Wireless Communications, 14(8) 2015, pp. 4277-4290.
[18] M. BERGAMO , R. R. HAIN, K. KASERA, D. LI, R. RANANATHAN, AND M. STEENSTRUP, *System Design Specification for Mobile Multimedia Wireless Network (MMWN)(draft)*, DARPA project DAAB07-95-C-D156, 1996.
[19] SANCHEZ, *Mobility Models*, http://www.disca.upv.es/misan/mobmodel.htm, page accessed on October 15th 2017.

[20] *K. H. Wang, and B. Li*, *Group mobility and partition prediction in wireless ad-hoc networks*, in Proceedings of Communications, ICC 2002, IEEE International Conference on. 2, IEEE, 2002, pp. 1017-1021.
[21] F. Bai, N. Sadagopan, and A. Helmy, *IMPORTANT: A Framework to Systematically Analyze the Impact of Mobility on Performance of Routing Protocols for Adhoc NeTworks*, in INFOCOM 2003, Twenty-Second Annual Joint Conference of the IEEE Computer and Communications, IEEE Societies, IEEE, 2 (March 2003), pp. 825-835.
[22] A. Mukherjee, N. Kausar, A. S. Ashour, R. Taiar, and A. E. Hassanien, *A disaster management specific mobility model for flying ad-hoc network*, Emergency and Disaster Management: Concepts, Methodologies, Tools, and Applications, 25, 2019, pp. 279-311.
[23] C. Zhao, M. L. Sichitiu, and I. Rhee, *N-body: A social mobility model with support for larger populations*, Ad Hoc Networks, 25, 2015, pp. 185-196.
[24] R. K. Sharma, and D. Ghose, *Collision Avoidance between UAV Clusters using Swarm Intelligence Techniques*, International Journal of Systems Science, 40(5) (2009), pp. 521-538.
[25] The VINT Project, *The network simulator ns-2*, http://www.isi.edu/nsnam/ns/, page accessed on 6 Jan 2016.

Edited by: Dana Petcu
Received: June 18, 2019
Accepted: September 9, 2019

AIMS AND SCOPE

The area of scalable computing has matured and reached a point where new issues and trends require a professional forum. SCPE will provide this avenue by publishing original refereed papers that address the present as well as the future of parallel and distributed computing. The journal will focus on algorithm development, implementation and execution on real-world parallel architectures, and application of parallel and distributed computing to the solution of real-life problems. Of particular interest are:

Expressiveness:

- high level languages,
- object oriented techniques,
- compiler technology for parallel computing,
- implementation techniques and their efficiency.

System engineering:

- programming environments,
- debugging tools,
- software libraries.

Performance:

- performance measurement: metrics, evaluation, visualization,
- performance improvement: resource allocation and scheduling, I/O, network throughput.

Applications:

- database,
- control systems,
- embedded systems,
- fault tolerance,
- industrial and business,
- real-time,
- scientific computing,
- visualization.

Future:

- limitations of current approaches,
- engineering trends and their consequences,
- novel parallel architectures.

Taking into account the extremely rapid pace of changes in the field SCPE is committed to fast turnaround of papers and a short publication time of accepted papers.

INSTRUCTIONS FOR CONTRIBUTORS

Proposals of Special Issues should be submitted to the editor-in-chief.

The language of the journal is English. SCPE publishes three categories of papers: overview papers, research papers and short communications. Electronic submissions are preferred. Overview papers and short communications should be submitted to the editor-in-chief. Research papers should be submitted to the editor whose research interests match the subject of the paper most closely. The list of editors' research interests can be found at the journal WWW site (`http://www.scpe.org`). Each paper appropriate to the journal will be refereed by a minimum of two referees.

There is no a priori limit on the length of overview papers. Research papers should be limited to approximately 20 pages, while short communications should not exceed 5 pages. A 50–100 word abstract should be included.

Upon acceptance the authors will be asked to transfer copyright of the article to the publisher. The authors will be required to prepare the text in LaTeX 2ε using the journal document class file (based on the SIAM's `siamltex.clo` document class, available at the journal WWW site). Figures must be prepared in encapsulated PostScript and appropriately incorporated into the text. The bibliography should be formatted using the SIAM convention. Detailed instructions for the Authors are available on the SCPE WWW site at `http://www.scpe.org`.

Contributions are accepted for review on the understanding that the same work has not been published and that it is not being considered for publication elsewhere. Technical reports can be submitted. Substantially revised versions of papers published in not easily accessible conference proceedings can also be submitted. The editor-in-chief should be notified at the time of submission and the author is responsible for obtaining the necessary copyright releases for all copyrighted material.