Flinders UNIVERSITY
ADELAIDE · SOUTH AUSTRALIA

Performance Improvement for Distributed Adaptive Data Processing of Flexible-sized Mobile Devices Network

by

Rui Li, *B.Sc.(Comp.Sc)*

Thesis
Submitted to Flinders University
for the degree of

Doctor of Philosophy
College of Science and Engineering
16 March 2018

Contents

List of Figures

List of Tables

List of Algorithms

Abstract

With the prevalence and convenience of current mobile devices, harnessing the combined resources of a number of mobile devices with limited computing capability to complete computing-intensive tasks is possible, especially in scenarios which lack powerful computing devices.

This thesis focuses on the organisation of mobile devices for a collaborative computing-intensive data processing task from the following perspectives: improvement of performance over slow wireless networks, task deployment strategies, and privacy preservation strategies. The network of mobile devices which is organised to collaboratively process computing-intensive tasks is named Distributed Adaptive Data Analysis Network of Flexible-sized Mobile Devices (FlexMNet).

This research proposes a strategy of predicting the optimised combination for which mobile devices should be selected to form a task-processing network. Research on the organisation of devices is presented in a proposed framework (FlexMNet). The framework provides a platform to make application (app) or plug-in development for mobile devices possible in the future. Privacy concerns within the framework have also been discussed with a proposed privacy protection strategy suitable to the framework.

Certification

I certify that this thesis does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person except where due reference is made in the text.

Rui Li

Adelaide, 20 June 2017

Acknowledgements

My great gratitude goes to my supervisor Dr. Denise de Vries for her professional supervision and caring friendship. I hardly believe that I am able to complete my Ph.D without her constant encouragement during this long journey.

I would also like to thank my co-supervisor Professor John Roddick for giving me the opportunities to undertake my study in Flinders University and the advice, support at the beginning of my candidature.

My heartfelt thanks go to my husband Dr. Qinyong Mao, who shared his most patience, cares and support during my candidature. Also, thanks go to my parents and parent-in-laws for their understanding and trust.

I believe I owe each person thanks in the School of CSEM at Flinders University who helped me in one or many other ways. In particular, Dr. Carl Mooney, Associate Professor Paul Calder, for their constructive criticism and discussions; Mrs Jennie Brand, for her assistance and encouragement when I need any help regarding policies and administration advice. My special thanks go to my dear friends Dr. Lua Perimal-Lewis, Dr. Adham Atyabi, Dr. Brianna Martin and Dr. Mark Reilly for their academic criticism, proofreading suggestions and life support throughout my candidature.

I appreciate Flinders University provides support, advice, resources and the staff training courses which improve my comprehensively problem solving skills.

Rui Li

June 2017

Adelaide.

Chapter 1

Introduction

Computing intensive data processing tasks, such as mining big data and meteorology programs, normally need high performance computing devices supported by complicated system architectures. Therefore, these tasks are normally executed by powerful computing facilities such as computer clusters and computing grids. However, these facilities are not cost-effective and therefore are often owned by large enterprises and research institutes, not by individuals and small-scale organisations. If data owners who do not own these facilities need to run a computing-intensive task on their data, they have to send their data to the organizations which have the facilities. Consequently, data owners have to face issues of cost, security and privacy.

Since 2010, a voice has become stronger that wirelessly networked mobile devices may become another form of executing computing-intensive tasks. This voice has been supported by three trends; increased availability of mobile devices in public, upgraded hardware in mobile devices and improved wireless networks. This voice have also been supported by an increasing number of research studies (Parmar, Jani, Shrivastav & Patel 2013, Comito, Falcone, Talia & Trunfio 2017).

The continued development of mobile devices, such as smartphones, tablets, wearable devices and laptops, over recent years has seen a significant transition towards wireless mobile computing. As shown in Figure 1.1 from Grand View Research (2015), smartphones are expected to represent over 60% of overall revenue in the personal electronic market by 2020 as a result of due to product innovation.

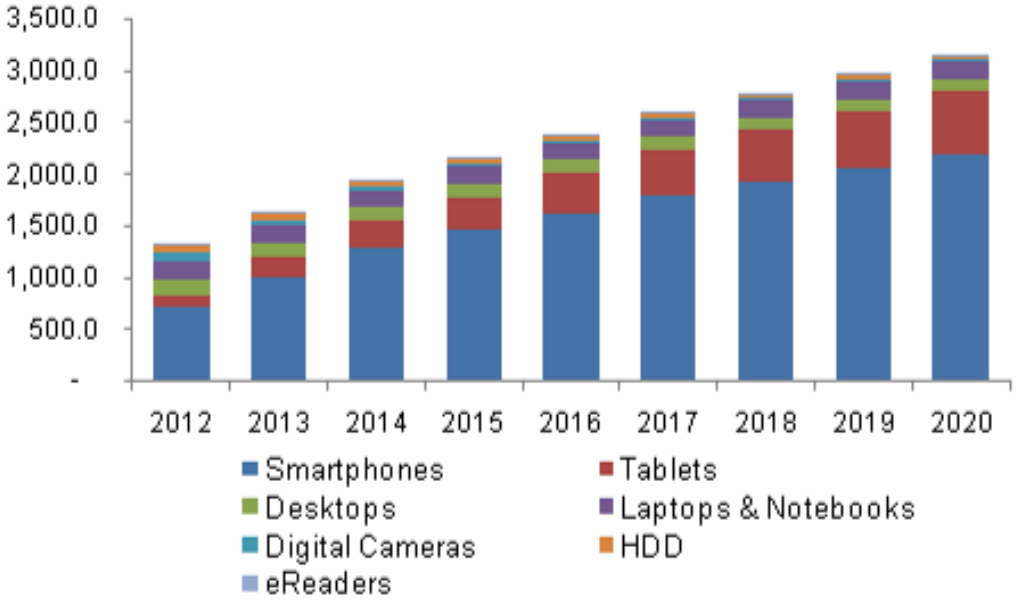

Figure 1.1: Global personal electronics market by product, 2012 - 2020 (Million Units) (Grand View Research 2015)

Additionally, these devices are equipped with powerful processors and also large memory spaces. In 2014, a majority of smartphones were equipped with 1 GB RAM and 2 GHz processor. By 2017, most smartphones in market can provide more efficient performance with hardware specifications of 4 GBs RAM and 2.3 GHz quad-core processor.

Table 1.1 lists the system specifications for contemporary prevailing personal mobile devices in the 2016 market sold on Amazon.com, including smartphones and laptops with wireless connection cards. The trend of mobile devices equipped with more advanced hardware is still moving upwards.

Apart from the greatly improved computing and communication capabilities of mobile devices, wireless network infrastructure which connects these mobile devices has been greatly improved in recent years. Wireless network infrastructure has been improved in the last decades which allows mobile devices to smoothly communicate and transfer a large amount of data in a reasonable amount of time. Wireless communication has greatly increased within various kinds of networks. The latest report presented by

Table 1.1. System specifications for contemporary prevailing personal mobile devices in 2016 market

Product	System	Entry year	CPU	Storage	Connectivity
Apple iphone 6s	Apple A9 APL0898	2015	Dual-core, 1840 MHz, Twister, 64-bit	2 GB RAM 128 GB Build-in Storage	802.11a/b/g/n/ac Wi-Fi with MIMO; Bluetooth 4.2
Samsung Galaxy A3	Samsung Exynos 7 Quad 7578	2015	Quad-core, 1500 MHz, ARM Cortex-A53, 64-bit	1.5GB RAM 16GB Build-in Storage	802.11 b, g, n; Bluetooth 4.1
Dell XPS 13 Ultrabook	N/A_1	2015	2.30 GHz, Dual-Core, 3MB cache	8GB RAM, 1866MHz RAM Speed 256GB SSD	DW1820A 2x2 802.11ac 2.4/5GHz; Bluetooth 4.1
Lenovo Yoga 910 Notebook	N/A	2016	2.7-GHz, Dual-Core, 4MB cache	16GB DDR3 RAM	802.11ac 2x2 WiFi; Bluetooth 4.1

1: N/A indicates that the system installed in the product varies depending on a customer's decision

CISCO in 2016 that " mobile network (cellular) connection speeds grew more than 3-fold in 2016 compared to 2015. Globally, the average mobile network downstream speed in 2016 was 6.8 Megabits per second (Mbps), up from 2.0 Mbps in 2015 " (Cisco 2017). The report also predicts that the "average global mobile connection speed will surpass 20 Mbps and 4G traffic will be more than three-quarters of the total mobile traffic by 2021 " within the next 5 years.

Additionally, more and more wearable devices, which are another form of mobile device and which greatly improve health conditions and well-being for human beings, have been developed in recent years. They have been equipped with wireless connection and considerably faster processors which can make them perform a certain level of data processing in their devices instead of only collecting data, similar to the sensors found in other digital wearable devices twenty years ago.

The trend of improving computing and communication capability of mobile devices and wireless networks is still moving upwards.

The continuous advances in these above mentioned two areas: wireless networking and mobile computing technologies provide the possibility of processing computing-intensive data by a group of mobile devices such as smartphones and tablets. Some applications and research on computing-intensive data processing use mobile devices to collect data, and then pass the data to a powerful data processing center, such as a computing cloud or a supercomputer.

Utilising networked mobile devices for computing-intensive data processing is significantly beneficial to a number of situations, including;

- Being small-medium business ventures, apart from the privacy concerns for their data, the costs of giving such applications to a third party organization is prohibitive, so owners are seeking other approaches with more cost-effectiveness and less risk to their sensitive data;
- In the situation that common mobile networks fail to work, or in the areas of no mobile networks, using mesh networks could enable the mobiles devices to communicate directly with one another.

The advantages of networked mobile devices have been noticed by researchers. The

development of CPU speed and storage capacity for laptops are typically behind for desktop PCs. However, Phan et al.(2002) argued that the potential aggregate power of these mobile devices is compelling for computing-intensive tasks in terms of three trends they observed.

1 CPU performance of a mobile device will grow according to the Moore's Law of increasing the density of transistors.

2 Wireless communication technology will grow for both local networks (for example, 802.11 or Bluetooth) and global networks (for example, 3G technology or ad hoc mesh. In 2017, 4G technology are available in most of urban areas in Australia).

3 Cooperative applications on grid will become common for computing-intensive tasks.

Therefore, Phan et al. firmly believe that mobile devices are the next generation of computing devices, due to their continuously upgrading computing and communication capabilities.

However, using mobile devices for computational purposes in research and in the implementation of processing computing-intensive data in wireless networks is still rare. The major reason for this is the low performance of these mobile devices.

The performance can be affected by several factors, such as battery constraints of mobile devices, limited capability of mobile devices compared to powerful computers, instability of wireless network infrastructure used to connect mobile devices, inefficiency of organization schema for a group of mobile devices and mobile users' privacy concerns.

As a result, this research analyses the factors that affect the computing-intensive data processing performance which further hinders mobile devices from being major computational devices in a computing-intensive data processing task. Chapter 2 reviews the previous research from the perspectives of wireless networks and mobile devices. In order to achieve the goal of improving performance of distributed data processing over a network of mobile devices, contributions are presented in chapters; Chapter 3 proposes a prediction strategy, Chapter 4 discusses the key factors affecting performance

and Chapter5 demonstrates an experiment in terms of the contributions in Chapters 3 and 4; Chapter 6 presents the contribution from the perspective of organisation of the involved devices; Chapter 7 presents the consideration from the perspectives of privacy protection; finally Chapter 8 concludes the work in this thesis by providing a structure of it in a diagram and analyses the future work.

Chapter 2

Literature Review

Part I. Wireless Computer Network

A computer network allows networked computers or digital devices to communicate with each other or exchange data. There are two types of networks in terms of the media that connects network nodes. These are *wired network* and *wireless network*. Wireless networks provide support for data transmission and communications for network nodes, such as mobile devices while data are processed over them. Part I reviews the infrastructure of wireless computer networks and its support for applications such as data processing.

2.1 Computer Networks

Computer networks comprise *network nodes* over *network links* in order to exchange data among the nodes (Shinde 2009). *Network nodes* are networked digital devices that send, route and receive the data. *Network links* are the cable or wireless connections between the nodes (Tanenbaum & Wetherall 2011).

Computer networks support a wide range of services and applications, such as share storage spaces and computing resources, access to the Internet, email and e-commerce platforms, and exchange instant messages, as well as many others.

According to the infrastructure of networks, a network can also be categorised into two major classes:

- *Wired network.* A wired network is simply a collection of two or more computers, printers or other digital devices connected by physical Ethernet cables. It is commonly used to transmit data between various digital devices and computer systems. Wired networks can also be used as part of other wired and wireless networks (Srividya & Vijayarani 2015).
- *Wireless network.* This computer network uses radio waves to connect network nodes or devices. Communications and accessing applications and information over the network without wires (Umar 2004). Different types of wireless antennas are used to send and receive the electromagnetic waves without using any physical conductors.

A brief comparison of the wired and wireless network has been done in (Kaur & Monga 2014).

Computer network books introduced a large number of concepts and terminology from the perspective of the layers of a network's architecture by using different approaches, such as a top-down approach (Kurose & Ross 2013) and a bottom-up approach from the layers of the architecture (Shinde 2009, Tanenbaum & Wetherall 2011). The following list sets out several basic concepts selected from these books which are relevant to transmission over networks.

- **Network structure** refers to the hierarchy of interconnected nodes within a computer network.
- **Network nodes** normally refer to individual digital devices that are networked and can send, route and receive data. Examples of nodes are personal computers (PCs) , servers and, printers, as well as network equipment such as routers, modems and hubs (Geier 2001). Two such digital devices are regarded as *being networked* when one device is able to communicate with the other one regardless of a *direct* connection between them. *Node* can have a specific definition in different computer networks. In a data communication network, a physical network node can be a data communication equipment such as a modem, router, bridge or switch. In a distributed network, the nodes normally refer to *clients*, *servers* or *peers* (Coulouris, Dollimore, Kindberg & Blair 2011). In data processing context, they are also called *computer nodes* or *computing nodes*.
- **Network links** refer to the transmission media connecting network nodes. These are often known as *network channels*, *communication links* or *data links*.
- **Three major performance properties** are *latency*, *transmission rate* and *throughput*, which affect the performance of applications supported by networks. Dif-ferent computer networks offer different *latency*, *throughput* and *transmission rate*.
- **Latency** is the time required to perform some tasks or process amount of data between the node which sends the data and the node which receives the data. It includes any kinds of delays typically incurred during processing data over the network, such as queuing delay, transmission delay and propagation delay (Kurose & Ross 2013). It is a significant element that contributes to network performance. A network that requires low latency connections normally expects short delay time.
- **Throughput** refers to the actual amount of data executed or results produced per unit of time. *Bandwidth* and *throughput* represent the capability of a network link from a (respectively) theoretical and practical perspective. *Throughput* is another critical performance measure in computer networks apart from *Latency*

(Kurose & Ross 2013). Protocol, data loss and, latency may be reasons that effect *throughput*.

- **Transmission rate** refers to the transmission of data at different rates over different network links. The transmission rate of a link is often measured in bits per second (bps) or bytes per second (Bps, 1 Bps = 8 bps).
- **Bandwidth** refers to the maximum amount of data that a network link can carry in a certain period of time, which is often regarded as the maximum transmission rate of a network. Bandwidth is often expressed as the number of bits (either kilobits or megabits) that can be transmitted in a second (Price 2007). Theoretically, the bandwidth of a network connection is normally fixed. However, the bandwidth that users experience varies due to being affected by latencies.

A network can be featured by its organisational purpose (Ding 2010, Wu & Irwin 2013).

- *Local area network*. A local area network (LAN) connects digital devices within a limited geographical area such as an office building, a home, a hospital, or an educational institutions. These devices are network nodes.
- *Backbone network*, which is a computer network infrastructure that connect diverse networks (such as LANs) within the same building, across different buildings, or over wider geographical areas. It provides the equipment and technical support for these networks to exchange information. Performance and congestion over networks are regarded as critical factors to be considered when a backbone network is designed. However, backbone networks are still widely implemented because they have larger capacity than the individual networks that connect to them. For example, a backbone network can be implemented in an international company in order to network its branches located in different countries. The Internet is another example of a backbone network, which networks a group of Wide Area Networks (WANs) together.
- *Virtual private network*. Virtual Private Network (VPN) is a dedicated network which connects routers, links and a Domain Name System (DNS) infrastructure and often uses IP security protocol (IPsec) to secure its data transmission.

Strong security features of VPNs is one of the major reasons for applying them. IPsec can help to secure data transmissions in the network layer between the data sender and the receiver. IPsec is adopted by many institutions such as companies, government departments and non-profit organisations for creating their VPNs that operate over the Internet (Kurose & Ross 2013).

- *Enterprise private network*, which is a network that a single organisation sets up to interconnect its departments or offices in different locations (for example, a headquarters, branch offices, production sites, retailer shops, and mobile offices used by travelling salespersons) in order to share business information within their organisation.

Computer networks can be designed for a broad range of purposes, which are (Shinde 2009):

- Sharing computing resources such as data, storage space or processors;
- Distributing processing functions;
- Conducting centralised control and management for distributed systems;
- Reducing the effect of the dissimilarity between networked equipment and software;
- Assisting network users to acquire the maximum performance at a minimum cost; and
- Achieving more efficient outcome for the transmission of large volumes of data among long-distance locations.

These purposes also define the applications and services supported by computer networks. Applications have also been summarised from the perspectives of Business, Home and Mobile in (Tanenbaum & Wetherall 2011).

- *Business Applications* include resource sharing, communication among employees and Electronic Commerce (e-commerce).
- *Home Applications* include internet connection, social network, online entertainment such as IP television and game playing, home-based e-commence such as

home online shopping and ubiquitous computing such as viewing photos in a home projector sending from several nearby digital cameras.

- *Mobile applications* include any applications over a network of mobile devices. Those applications are normally carried out by fixed wireless network at home and in an office.

The literature review continues within the scope of wireless networks.

2.2 Wireless Networks

At present, one important network type is wireless network, due to the fast development of wireless technologies, such as cellular systems, radio and spread spectrum technologies, and free space optical communication technology (Tanenbaum & Wetherall 2011). Applications over wireless networks are enormous. One example is a mobile office. People on the road can use their portable digital devices to make calls, send and receive e-mails, read remote files, log in on remote machines, browse the Internet from a location that they prefer (Shinde 2009).

One advantage that wireless networks have is they are less expensive and much easier to install than more traditional wired networks. The transmitted distance over a wireless network varies between a few metres to thousands of kilometers in any locations. In some areas, where limited budget and geographic location restrict communication, specially designed wireless networks are essential for solving local communication problems. One example of this kind of wireless network is a *Serval Network* (Gardner-Stephen, Challans, Lakeman, Bettison, Gardner-Stephen & Lloyd 2013). A *Serval Network* is a telecommunications system that consists of at least two mobile phones which are able to work outside of regular mobile phone tower range based on the Serval App and Serval Mesh. This system assists people to communicate at any time, anywhere and privately with other people via smart mobile devices. Another example of a wireless network is a specially designed, low-cost wireless network for the developing world (WNDW.net 2013).

Wireless networks can be categorised into many types depending on the criteria chosen

for their classifications. Two criteria which are relevant to this research are *communication coverage area* and *network formation and architecture* (Basagni, Conti, Giordano & Stojmenovic 2004).

The types of wireless networks are normally discussed in terms of the distances over which data are transmitted and the radio spread spectrum of a network. These types include (Gordon 2015):

- *Wireless Personal Area Networks (WPAN)* refers to the network that connects the digital devices within a small scope and is generally created for the individual uses. Signals that covers the WPAN are provided by the technologies, such as bluetooth radio and invisible infrared lights. Since 2010, Wi-Fi signals are becoming commonplace in WPANs covered by as Wi-Fi signal connectors are integrated into a variety of digital devices by their designers and manufactures.
- *Wireless Local Area Networks (WLAN)* link mobile devices using wireless network infrastructures in order to support the mobile networking applications that need to get benefits in process efficiency, accuracy, and lower costs. As one of the important benefits of using WLANs, mobility implemented by the spread-spectrum or Orthogonal Frequency Division Multiplexing (OFDM) technologies enables users to move around while remaining connected to the network (Geier 2001).
- *Wireless Ad hoc Networks* are also known as *Wireless Mesh Networks* or *Mobile Ad hoc NETworks (MANETs)*. Traditional networks are operated in infrastructure mode, where the hosts associated with a base station which provides all services (e.g. address assignment, routing and DNS services). In ad hoc networks, wireless hosts do not reply on such infrastructure. Without such infrastructure, all services which are normally operated by a base station are provided by the hosts themselves. When the hosts are mobile, with connectivity changing among nodes, these networks are known as mobile ad hoc networks (MANETs) (Kurose & Ross 2013). Various network layer protocols need to be designed to realise ad hoc mobile networks. Examples of these protocols are *distance-sequenced distance-vector routing* (Perkins & Bhagwat 1994) and *dynamic source routing*

(Johnson & Maltz 1996).

- *Wireless Wide Area Networks (WANs)* are wireless networks that extend over large geographical areas, such as a CBD and its suburbs in a city, and a headquarter and its branch offices in an enterprise.
- *Global Area Networks (GANs)* are constructed by inter-connecting a number of WLANs which finally cover an unlimited geographical area. The Internet is regarded as a GAN. The key challenge of applying GAN is to deal with seamless communications among users from one WLAN to another (Ding 2010).

Several wireless networks which are popular used in data processing include the following types.

2.2.0.1 Distributed Networks

Distributed network is also known as *distributed network system*. It is a system consisting of a collection of communication networks and software systems designed to produce an integrated and consistent computing environment. One of the key purposes of the distributed networks is to share resources or communications (Jia & Wanlei 2005).

There are significant confusions in the literature about the difference between a distributed system and a computer network. A distributed system usually presents to the users as a single coherent model or paradigm although it is a collection of independent computers. A computer network usually presents to the users the actual machines because of the absence of the coherence, model or paradigm. Additionally, a distributed system has a software system built upon a physical network while a computer network does not have (Tanenbaum & Wetherall 2011).

2.2.0.2 Ad hoc Computing Networks

An ad hoc wireless network is a group of adaptively and wirelessly networked two or more devices equipped with the capabilities of communications and self-organisation. The computation, storage and communications capabilities are various in different

types of ad hoc wireless devices, for example, notebook, laptop, smartphone, wearable mobile device. Such devices can immediately communicate with another device that is within or outside their radio range. Due to the feature of being adaptive, a formed network can be de-formed at any time without a system administration support (Toh 2001).

2.2.0.3 Volunteer Computing Networks

In volunteer computing networks, the peers contribute to the solution of a computationally intensive problem by freely providing their computational resources, i.e., without seeking financial benefit. Three project examples of Volunteer Computing Networks are Bonic (*The BOINC project* 2017) and Simple Light-weight Infrastructure for Network Computing (SLINC) (Baldassari, Finkel & Toth 2006).

2.2.0.4 Sensor Networks

Wireless sensor networks are formed by micro sensors embedded in mobile devices. They are normally regarded as a special type of ad hoc wireless network (Toh 2001). However, recent research regarded ad hoc network and sensor network as two different research areas, such as most research presented in IEEE 9th and 10th International Conference on Mobile Ad hoc and Sensor Networks in 2013 and 2014. The micro sensors are normally small and equipped with communication and storage capabilities. They are also widely applied in science, military and healthcare to continuous monitor targeted information for further analysis.

2.3 Mobile Devices in Wireless Networks

A mobile network consists of mobile nodes and static nodes. Mobile nodes are devices that can be moved over a time period defined by users. Mobile phones are classic mobile nodes. Static ones remain static over a time period defined by users. Wireless access points are typical static nodes (Goh & Taniar 2005).

Wireless networking and mobile computing have a close relationship, however, they are not identical. Table 2.1 shows the distinction between wireless networking and mobile computing through several example applications (Tanenbaum & Wetherall 2011).

Table 2.1: Distinction between wireless networks and mobile computing (Tanenbaum & Wetherall 2011)

Typical applications	Wireless networking	Mobile computing
Desktop computers in offices	No	No
A notebook computer used in a hotel room	No	Yes
Networks in unwired buildings	Yes	No
Store inventory with a handheld computer	Yes	Yes

Smartphones combine with the aspects of mobile phones and wireless networking. Many current smartphones are able to connect to wireless hotspots, and automatically switch between networks in order to choose the best option of connection for their users. They transmit data by using the Internet and make phone calls through connecting to the 3G and 4G cellular networks. Since mobile devices were equipped with GPS (Global Positioning System) application, many GPS-enabled services became popular, such as mobile maps for searching a nearby bookstore or a restaurant (Tanenbaum & Wetherall 2011). Additionally, the development of sensor networks stimulates the applications of mobile devices. For example, small nodes in sensor networks wirelessly collect the information they monitor and transmit the information to a data center for analysis. These small nodes can be integrated in mobile devices, such as smartphones in cars, for gathering data on locations, water temperature, vibration, speed, and fuel efficiency from its on-board diagnostic system and upload the information to a database (Hull, Bychkovsky, Chen, Goraczko, Miu, Shih, Balakrishnan & Madden 2006). Those sensor data can also help drivers or users find potholes, plan trips around congested roads. The sensor networks combined with mobile devices are also required by scientific studies. One example is tracking themigration of individual zebras by placing a small sensor on each animal (Juang, Oki, Wang, Martonosi, Peh & Rubenstein 2002). Researchers also use these wireless mobile devices to track the life habits of small birds and insects (Warneke, Last, Lifebowitz & Pister 2001).

Another driving force of making mobile devices widely used is mobile device makers

and network operators for commercial reasons, which promotes m-commerce (mobile-commerce) (Senn 2000). When equipped with Near Field Communication (NFC) technology, the mobile can interact with a card-reading device for a payment acting as a Radio-Frequency IDentification (RFID) smartcard. For example, payments for shopping and entertainment can be authorised through the tap-and-go services installed in the mobile devices instead of cash and credit cards.

Wearable devices is another expanding application area of mobile devices. Most of them can be controlled over wireless networks, which can be configured more easily by doctors. These devices help doctors get real-time data for patients' healthy conditions and then make more precise decisions (Halperin, Heydt-Benjamin, Ransford, Clark, Defend, Morgan, Fu, Kohno & Maisel 2008).

These mobile devices are regarded as *smart objects* if they are " autonomous, physical digital object augmented with sensing/actuating, processing, storing, and networking capabilities " (Fortino & Trunfio 2014). These heterogeneous *smart objects* such as sensors, smart phones, RFID which are connected and conduct cooperative applications over networks, especially global area networks are called The Internet of Things (IoT). Studies on IoT based on these smart objects can be found in (Fortino & Trunfio 2014).

2.4 Summary of Part I

As one of the two major types of computer networks, wireless networks is experiencing a fast development with the improved wireless technology. Due to the advances in wireless networks, in order to complete a certain computing task, there is a possibility of using a number of connected mobile devices which are equipped with faster processors, bigger memory storage compared to ten years ago and intelligent functionalities. Grouping a number of mobile devices needs the support from the wireless computer network, therefore, Part I reviews the wireless network, especially concentrates on several types of wireless networks in which mobile devices are normally adopted. Part I also reviews the advances and and the difficulties of current wireless networks when

mobile devices are involved.

One of the applications over wireless network is data processing. Computing-intensive data processing attracts lots of interest from academia and industry. Mobile devices have been considered to get involved into computing-intensive tasks due to their up-to-date technologies in both hardware and software. Motivated by these, Part II reviews the studies in processing the computing-intensive data. As information that can be processed over distributed networks are various with different processing strategies, in Part II, data mining strategies are reviewed as an example for studying the issues of processing over wireless networks.

Part II Computing-Intensive Data Processing

Computer networks can provide support for various applications, such as processing data, social communications, instant messaging, video streaming, distributed games and peer-to-peer file sharing. Processing data, especially computing-intensive data over computer networks is the focus of this review. Processing Big Data is regarded as a procedure which requires intensive computing. In terms of this understanding, one common approach to processing Big Data, *data mining*, is reviewed in this part. The definitions of the computing-intensive data and different procedures to process the data are also reviewed.

2.5 Computing-Intensive Data

In the Oxford Dictionary, *data* represents 'quantities, characters, or symbols on which operations are performed by a computer and which are stored and recorded on magnetic, optical, or mechanical recording media'. It can be transmitted in the form of digital electrical signals. Digital data are digitally codified and often stored in relational databases, which are also called data set (Harrington 2016). Most commonly, a data set is stored in a database as a single relational table where every column of the table represents a particular attribute, and where each row represents the elements of one relation in the data set (Weiss 2016).

A data set is digitally stored in physical storage media. It needs to occupy a space in storage media measured by bytes. The *size* of a data set refers to the number of bytes that a data set occupies in a physical media. The same data set varies in size in terms of several factors, such as its data structure and the filing system it is stored in.

The volume of data has never been seriously discussed by academics until the concept of Big Data emerged at the end of 20th Century. The sudden rise of Big Data has left many unprepared, including the definition of Big Data (Gandomi & Haider 2015). While ubiquitous today, Big Data as a concept is promising and has uncertain origins. Diebold (2012) argued that the term " *Big Data . . . probably originated in lunch-table conversation at Silicon Graphics Inc. in the mid-1990s, in which John Mashey figured prominently.* ". Although definitions of Big Data from different perspectives have raised some confusion, the widely accepted dimensions for defining Big Data are Volume, Variety and Velocity (or the Three V 's). 'Volume' refers to the magnitude of data. 'Variety' refers to the structural heterogeneity in a data set, including structured, semi-structured and unstructured data. 'Velocity' refers to the rate at which data are generated and the speed at which the data should be analysed (Laney 2001, Chen, Chiang & Storey 2012, O. Kwon 2014). This thesis focuses on the dimensions of Volume and Variety.

2.6 Computing-Intensive Data Processing Strategies and Procedures

Data processing is one of the applications that can be carried out over a wireless network which generally collects and manipulates items of data to produce meaningful information. The field of data processing has some overlaps with the fields of communication as both need to transmit/exchange information over wireless networks, although they use different information processing strategies.

In terms of the organisation of processing networks, strategies which are used to process computing-intensive data include two approaches;

- *Centralised approach*, which indicates that the processing task is executed by a single but powerful computing device or system, such as a supercomputer or a cluster of multiprocessors (CLUMPs). The algorithms used to process the data are normally centralised and parallel algorithms (Buyya 1999, Prabhu 2008).
- *Distributed approach* which indicates the processing task is executed by a group of geographically distributed computing devices that have less computational capability compared to high performance computing devices. The group of devices collaboratively conduct the data processing workload. The algorithms used to process the data are normally designed in a distributed manner (Attiya & Welch 2004, Datta, Bhaduri, Giannella, Wolff & Kargupta 2006, Zubairi 2009).

The connections between computing nodes are established using either wired technology or wireless technology (Wysocki, Dadej & Wysocki 2005). Accordingly, processing data using the distributed approach are classified into two groups: wired data processing and wireless data processing. Data processing over wireless computer networks, e.g. sensor networks or mobile networks, has attracted research interest for many years. Mobile devices have been organised as a network to process data (Imielinski & Korth 1996, Umar 2004, Sinha, Ghosh & Sinha 2016). This data processing network benefits from improved wireless network and communication technologies and the advantages of current mobile devices, which are;

- their widespread availability in public;

- their mobility making them feasible for accessing to networks;
- their low cost compared with powerful computers; and
- their highly improved computational capabilities compared to a decade ago.

On the other hand, in some situations, data owners cannot use the centralised approach due to high cost, inconvenience, personal considerations or other limitations. Consequently, the distributed approach (the network of computing devices strategy) is considered to be an option to deal with these issues. With the prevalence of mobile devices, they are now considered as being able to participate in the computing network. The computing capability of an individual mobile device may not be powerful. However, a number of these devices can be grouped or networked as a computing service provider, which has the following advantages (Huerta-Canepa & Lee 2010);

- they are becoming more common, which suggests an increasing availability of nearby devices to form a computing network;
- over time they being equipped with more powerful computing capabilities;
- they integrate various network interfaces which allow mobile devices to communicate with each other with a minimum financial cost; and
- they allow for the creation of communities in which shared tasks can be executed.

Apart from the advantages listed above, in some situations, considering an existing or a self-constructed network of mobile devices to process data is essential. For example, in areas where there is no connection to either an internet or a computing Cloud, mobile device owners need to process their data, even though their own devices do not have the capability to process it. One possible solution to this problem is that these devices form a temporary computing network.

Common data processing strategies include *statistics* and *data mining*. In both academia and application fields, data mining is a more commonly used strategy for processing Big Data set than statistics.

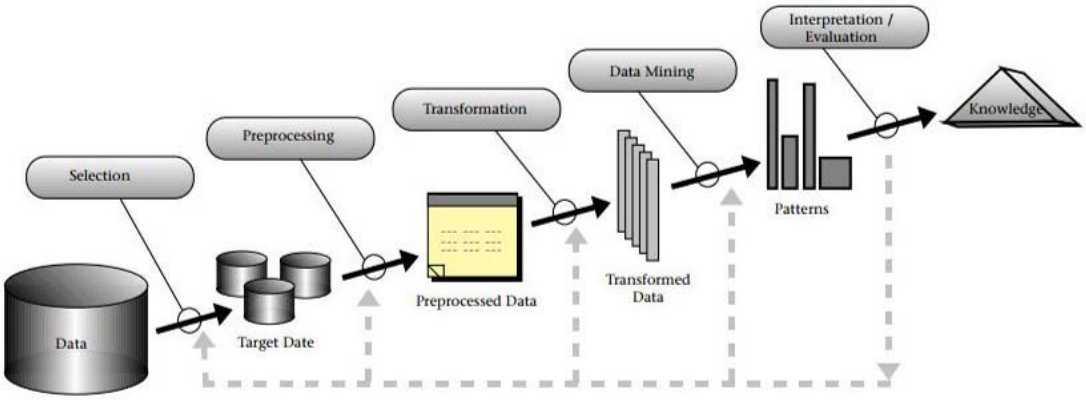

Figure 2.1. An overview of the steps in KDD (Fayyad et al. 1996*b*)

2.7 Data Mining: A Data Processing Approach for Big Data

Data Mining (DM) is a data analysis step in the *Knowledge Discovery in Database (KDD)* process for searching interesting and unknown patterns in a particular representational form from large data sets (Fayyad, Piatetsky-shapiro & Smyth 1996*b*). KDD is the non-trivial process of eliciting interesting knowledge from potentially very large data repositories. The commonly accepted knowledge discovery architecture is presented in Figure.2.1. From data to knowledge, several steps have to be taken, including selecting and cleaning data inputs, extracting features on the predominantly cleaned, processed data, mining the data by using DM algorithms, and outputting patterns and relationships for interpretation and evaluation (Fayyad, Piatetsky-Shapiro & Smyth 1996*a*).

DM, as the major step in KDD, is a discipline situated at the interface of a range of disciplines including, but not limited to (Jackson 2002);

- statistical analysis;
- neural networks and genetic algorithms;
- pattern matching; and

- intelligent machine learning.

Generally, the DM process has two models defined by the two approaches for processing computing-intensive data;

- *Centralised mining process*, which adopts data integration methods to gather all data into a centralised site, then run a mining algorithm over the centralised data set to extract useful patterns and trends. The mining algorithm handles all computations during the mining process. This model is widely adopted by traditional DM technology. In this model, raw data are pooled together for mining purposes.
- *Distributed mining process*, which mines data sets which are not geographically stored in the same site. There are a number of issues arising when a distributed mining process is running, including storage cost, communication cost, computation cost and privacy concerns from data owner(s).

The difference between these two processes is whether the DM task works on a (physically or virtually) distributed data set or a centralised data set.

2.7.1 Distributed Mining Process

Distributed mining processes are studied in two large subfields in DM: *Parallel Data Mining (PDM)* and *Distributed Data Mining (DDM)*. As there is a significant overlap between the two fields, researchers also study them as in one field called *Parallel and Distributed Data Mining (PDDM)*.

PDM uses high-performance architectures to deal with tightly coupled systems including Shared-Memory Systems, Distributed-Memory Machines, or clusters of Symmetric MultiProcessing (SMP) workstations (Zaki & Ho 2000). It studies parallel algorithms, methods, and strategies for discovering novel and useful patterns from large data sets. When DM techniques are implemented on such high-performance parallel machines or workstations, large data sets can be analysed in a more reasonable time which is more practical for users. Additionally, faster processing makes users experiment with more models for their data in a limited time period, which helps users understand complex data. The main goals of introducing parallel computing technologies

in DM are (Congiusta, Talia & Trunfio 2005):

(a) improving the performance of existing techniques;

(b) implementing innovative parallel techniques and algorithms; and

(c) concurrently analysing and integrating results generated from several DM techniques in parallel in order to get a better model such as in accuracy.

PDM is a potential solution for organisations with centralised data stores although it has many challenges including minimising synchronisation and communication, minimising I/O cost, minimising duplication of work, balancing computing loads, and decomposing data efficiently (Albashiri 2013). Reviews regarding PDM include its system and algorithms (Zaki 2000, Zaki & Pan 2002), a particular application on Hadoop Map Reduce (Lokeswari & Jacob 2016) and a cooperation with a Cloud service (Kholod, Kuprianov & Petukhov 2016). However, the overheads of computing facilities, such as clusters of SMP workstations, is still unaffordable for most organisations which lack strong financial support. In addition, when the data resources are geographically distributed or when the data which are owned by more than one custodian are not sharable because of the privacy issues, PDM is not the appropriate technique to deal with those situations. Due to these privacy concerns, the inability to pool distributed data into a single site leads the research direction in DDM to the topics of communicating intelligence over data sources.

Strategies exploiting parallelism on DM algorithms can be identified in three groups:

(a) Independent parallelism (e.g. a parallel implementation of the C4.5 algorithm (Kufrin 1997));

(b) Task parallelism (e.g. data distribution for parallel Apriori (Agrawal & Shafer 1996)); and

(c) Single program multiple data (e.g. parallel version for the AutoClass algorithm, P-AutoClass (Foti, Lipari, Pizzuti & Talia 2000)).

DDM deals with loosely-coupled systems such as a cluster of computers over a slow LAN. It also deals with geographically distributed sites over a WAN such as the Internet (Zaki & Ho 2000). DDM originated from the need for mining distributed

data sources, which includes two major tasks; (1) processing local data at each distributed site; and (2) fusing the local knowledge in order to discover global knowledge at a global level. It concerns the study and application of classical DM procedures in distributed computing environments and of making the best of the available resources (communication network, computing units and databases) for the environments (Fu 2001, Park & Kargupta 2003).

The major motivations for applying DDM are (Congiusta et al. 2005):

(a) It is an alternatively scalable solution for distributed data sets which can also be processed by centralised mining algorithm;

(b) Performance could be improved for a large centralised data set;

(c) Distributed data sets can be gathered for processing in terms of various of human factors (for example, privacy concern) and technical reasons (for example, data are untransferrable because of it exists in the excessive amounts or the digital uninterpretable format of data set); and

(d) In order to get better results such as in accuracy, a concurrent analysis could be applied with result integration by using different DM techniques over each distributed data set.

DDM is essential in the situations that data sets are stored in a number of distributed locations. DDM studies the cases in which data sets that need to be mined are distributed for reasons of geography or privacy. It is an ideal way for the organisations where data owners have insufficient computing facilities to satisfy their DM requirements. Grid-based DM has many applications for these organisations. A grid refers to a geographically distributed computation infrastructure which allows users to access the resources stored in a number of heterogeneous computing machines associated to the infrastructure. The grid is regarded as the extension of the distributed and parallel computing paradigms. Due to the properties of the scalability and heterogeneity, grid environments are suitable for both computing-intensive tasks and data-intensive tasks. A group of researchers conducted a number of studies in extracting the useful knowledge or patterns from large databases on grids (Cannataro, Congiusta, Pugliese, Talia & Trunfio 2004, Trunfio, Talia, Papadakis, Fragopoulou, Mordacchini, Pennanen,

Popov, Vlassov & Haridi 2007, Congiusta, Talia & Trunfio 2008).

PDM and DDM have many differences and communication cost and data distribution are regarded as the main ones. Zaki and Ho (2000) also explained their understanding of the differences. They regarded PDM is an essential component of a DDM system, namely, multiple PDM sites constitute a DDM system. These PDM sites can be a single workstation, a cluster of SMPs or a supercomputer. On the other hand, they regarded DDM as a collection of parallel machines or PDM systems which are either tightly-coupled or loosely-coupled. However, they also mentioned that it is hard to draw a line between the two areas because there is a significant overlap between them.

The contributions of DDM in the last twenty years have been reviewed by several papers (Park & Kargupta 2003, Tsoumakas & Vlahavas 2008, Zeng, Li, Duan, Lu, Shi, Wang, Wu & Luo 2012, Rafrastara & Deyu 2016) since DDM became a major research direction in DM.

To summarise, PDDM is a broad research direction, which has been further redirected to multiple new research directions for specific application needs. PDDM strategies for several wireless networks are reviewed in the following sections, including;

- Data mining over peer-to-peer networks
- Data mining over wireless sensors networks
- Data mining over ad hoc networks
- Data mining over multi-agent based networks
- Data mining over computing Clouds

2.7.1.1 Data mining over peer-to-peer networks

Peer-to-peer (P2P) networks are created by two or more connected computing nodes, which can share resources among these nodes without going through a separate server computer. It is characterised by an asynchronous nature, limited battery power, weak communication and computational capabilities, and the existence of faults (Bandyopadhyay, Giannella, Maulik, Kargupta, Liu & Datta 2006).

P2P networks gain popularity in DDM because they can deal with the issues of data analysis in situations that data resources, computing sites, and users are distributed. The algorithms for P2P systems have dramatic difference with those for traditional centralized DM algorithms. They requires more features in scalability, availability, asynchronism, decentralization, fault tolerance, privacy and security. A large amount of communication among computing nodes required by many DDM tasks is not suitable for P2P networks because of being unlikely to scale the task, which causes low performance. One approach which may solve the problem is that eliminating the need for an extensive communication load. For example, to compute the distance matrix (in some metric space) where the $(i, j)_t h$ entry represents the distance between tuples stored at the $i_t h$ and $j_t h$ nodes, an efficient P2P algorithm can be designed by identifying only the significant entries of the distance matrix and without exchanging information between every pair (Datta et al. 2006). Apart from the performance, the other issue that DM algorithms in P2P networks are facing is how to quantify an algorithm would behave over a given finite amount of time.

To improve the performance of processing tasks over P2P networks, the cutting-edge technologies, such as grid computing, high-performance computing have been introduced. However, issues on distributed computing such as limitation on data transmission bandwidth, skew distribution, very-large data size, privacy preservation are still remaining (Zeng et al. 2012). On the other hand, no other research discussed the performance when mobile devices are involved in P2P networks.

2.7.1.2 Data mining over wireless sensors network

Wireless Sensor-based Networks (WSNs) is one type of P2P network. Features of WSNs are featured by wireless sensor nodes with lower power requirements, easy of deployment and the support of a large number of sensors distributed over a wider area.

As sensors become more inexpensive and more easily deployable, mining interesting patterns directly from the raw and noisy sensor data over WSNs is becoming popular. Embedded WSNs, consisting of small, low-power devices carrying fast processors,

memory, and wireless communication, could play an important role in these DM requirements. WSNs are being deployed for applications such as habitat monitoring (MainWaring, Polastre, Szewczyck, Culler & Anderson 2002), human activity recognition (Gu, Wang, Wu, Tao & Lu 2011), volcanic activity monitoring (Allen, Johnson, Ruiz, Lees & Welsh 2005), and location tracking systems (Kung & Vlah 2003). Studies on applying traditional DM strategies to WSNs can be found in (Duarte & Hu 2004, Gummadi, Li, Govindan, Shahabi & Hong 2005). A technique for clustering homogeneously distributed data in a P2P environment like a sensor network has been proposed (Bandyopadhyay et al. 2006). This algorithm studies the extensive theoretical analysis of the K-means algorithm by bounding the error in the distributed clustering process.

Outlier detection and data fusion have been regarded as two major difficulties for DM over WSNs.

Outlier detection has been highlighted in WSNs because of several factors. First, the data are collected through imperfect sensing devices. Second, a sensor's performance tends to deteriorate when the battery power of a sensor is exhausted. Third, error occurrence accumulates because of a large number of sensors in these networks. Finally, sensors are prone to gaining the attention of competitors who can interfere with them. In summary, outlier detection should be considered in any data processing procedure in WSNs. Meanwhile, Branch et al. (2006) proposed a methodology that solves the problem of unsupervised outlier detection in WSNs. Their solution proposed the following features;

(a) allowing flexibility in the heuristics used to define outliers;

(b) working in networks with communication load proportional to the outcome;

(c) being robust with respect to data and network change; and

(d) revealing the outcome to all of the sensors.

Data fusion is another inseparable research direction in DM over WSNs, the task of which is to integrate from separate sources with minimum information loss. Wu et al. (2004) proposed a distributed data fusion approach for WSNs that integrates disparate data sources. Their approach includes two steps: firstly each sensor makes a local

decision; secondly all local decisions are combined at a fusion centre to produce a global decision. The objective of this approach is to maximise the probability of signal detection through determining the optimum local and global decision rules. Typically the decision requires hypothesis testing techniques. The Bayesian and the Neyman-Pearson criteria are often used for this purpose (Park & Kargupta 2003).

Wireless Sensor Data Mining Project (WISDM) demonstrate an application that employs WSNs and distributed mobile DM (Fordham University & Science 2012). This project uses DM methods on accelerometer data to identify the activities that smartphones and other powerful mobile device users are performing (activity recognition) while carrying the devices. A tri-axial accelerometer is the major sensor in these devices. A major sensor in these devices was originally included for screen rotation and advanced gaming.

2.7.1.3 Data mining over ad hoc networks

A framework has been proposed to allow ad hoc DM queries to mine association rules within a data warehouse (Nestorov & Jukic 2003).

To support data-intensive analysis activities and processes, an Ad hoc Data Grid Environment (ADAGE) framework is proposed to satisfy all requirements and overcomes the limitations of the two current design approaches: horizontal and vertical. The horizontal approach focuses on a generic system design that can easily be extended through new modules or services. With more new functionalities added over time, the system's complexity increases and incompatibilities arise. The vertical approach focuses on a system design with restricted scope. Unfortunately, the scope itself also limits the system's ability to be extended beyond that scope (Yao 2013).

2.7.1.4 Data mining over multi-agent networks

Integration of intelligent agents and DM in a mining system generated another research direction: Multi-agent based distributed Data Mining (MaDM). Constructing DM systems from agent perspectives or integrating agents into DM systems can greatly improve the flexibility of DM systems (Zhang, Zhang & Cao 2005).

Integrating DDM with Multi-Agent Systems (MAS) for computing-intensive applications has been appealing because the distributed agents in MAS are often proactive and reactive. The architecture of MAS has been adopted by most DDM systems.

In a distributed environment, a DM task can be executed in parallel and information can be exchanged between a set of agents. These agents are equipped with an algorithm for dealing with given data sets and mining tasks. However, concurrency issues arise. A MaDM system is proposed for handling this issue (Moemeng, Gorodetsky, Zuo, Yang & Zhang 2009). In the data layer of this system, data are hosted in various forms, such as online relational databases, data streams, web pages, etc. In the communication layer, through communication protocols, the system communicates to a system component, Directory Service which stores the list of data types, mining algorithms, and data sources. Another system component, Query Optimizer, is used to analyse the request to determine type of mining tasks and choose appropriate resources for the request. It also makes decisions on whether it is possible to parallelise the tasks. The Discovery Plan in MaDM allocates sub-tasks with related resources.

Ontology-based integration of agents and DM from a theoretical perspective has been discussed in (Zhang et al. 2005).

In advanced MaDM, an agent is expected to be intelligent and autonomous, and can also exchange intelligence with other agents. The *intelligence* used in MAS is based upon social behavior, not the metaphor of individual human behaviour in classical Artificial Intelligence (AI). It emphasises actions and interactions, complementing knowledge representation and inference methods in AI. In MAS, an *intelligent agent* is defined as a software entity that performs specific tasks on behalf of users with varying degrees of autonomy and intelligence (Honava, Miller & Wong 1998).

2.7.1.5 Data mining in Cloud computing

Cloud computing was proposed by several commercial companies, including Google and Amazon in 2006. The core part of a Cloud computing service is a data centre. Cloud computing combines several traditional technologies including grid computing, distributed computing, parallel computing, utility computing network storage tech-

nologies, visualisation, load balance. The key word in Cloud computing is *virtual*. When users use a Cloud computing service, they do not have to know the location of the service and how the service has been allocated to them. Users tell the service their needs and the service accomplishes the rest of the task with the user-defined outcomes returned (Weiss 2007, Buyya, Yeo & Venugopal 2008).

The similarities and differences between these two services, Cloud computing and grid computing, were presented in (Hashemi & Bardsiri 2012). Cloud computing is regarded as a model which gives on-demand and convenient network access to get benefit from a shared computing resources pool including networks, servers, storage, applications, and services and can be regarded as the intelligent extension of traditional *grid computing*. The significant similarity of these two services is that they both aim to construct a powerful computing capability. On the other hand, the major difference between these two is significant; Cloud computing pursues more stable and rapid storage, while *grid computing* aims for the powerful computing capability.

The features of the powerful computing capability and the virtual nature of Cloud computing significantly benefit DM tasks which normally require large computation workloads. Research on creating systems on the combination of these two technologies is popular. Kholod et al. (Kholod et al. 2016) studied the possibility of creating a Cloud system for debugging parallel and distributed data mining algorithms as an analytical service. Other research proposed strategies for building platforms on their combination in application scenarios, such as, students examination system (Li, Wang & Gao 2013) , electric power system (Zhu 2017), and in healthcare system with the integration of clinical and genomics data (Anjum, Aizad, Arshad, Subhani, Davies-Tagg, Abdullah & Antonopoulos 2017).

However, the issues arising in Cloud computing have also attracted attention, including security, personal data privacy and legal issues, such as regulatory control, competition and cross-border regulation (Cheung & Weber 2015). Additionally, as Cloud computing is service-oriented, whether the cost of using the service is affordable by an organisation or an individual is under constant discussion (Dillon, Wu & Chang 2010).

2.7.2 Mobile Devices for Data Mining

Mobile devices normally adopt DM strategies to process the data stored or generated by them. Most of the current DM techniques are used by third party provided computing services, such as computing Cloud or supercomputers to process mobile data (Gaber, Stahl & Gomes 2014, Khan & Sivrikaya 2015). Data generated from mobile devices are sent to a computing Cloud for processing computing-intensive tasks. However, mobile devices themselves, although with limited computing and storage capabilities, could be organised to undertake computing-intensive processing activities.

However, since 1996, Gray et al. (Gray, Kotz, Nog, Rus & Cybenko 1996) have explored the possibility of using mobile agents for mobile computing. Also, mobile devices have become involved in DM with the prevalence of smart mobile devices since 2005. For these reasons, there is a possibility that when a number of mobile devices are networked, they can collaboratively do certain tasks *for* DM, instead of *with* DM. As computing-intensive tasks normally occupy processor and storage, as well as being time consuming, studies of energy/battery saving and fast and efficient data processing algorithms/strategies have been highlighted. Research on energy saving can be found in (Pluntke, Eggert & Kiukkonen 2011, Vallina-Rodriguez & Crowcroft 2011). Ariwa et al. (2003) proposed an e-business model application using a mobile agent for DDM. A distributed and mobile DM system has been proposed in (Wang, Helian, Guo & Jin 2003).

WSNs are regarded as a special type of distributed mobile network because of the mobility of networked sensors. Hull et al. proposed a distributed mobile sensor computing system (Hull et al. 2006). An efficient mobile DM model has been proposed in (Goh & Taniar 2005).

With the significant growth of mobile applications and the arising of the Cloud computing concept, Mobile Cloud Computing (MCC) has been introduced as a potential technology for mobile services since 2010. MCC integrates mobile devices into Cloud computing environment and needs to overcome difficulties in performance (for example, battery life, storage, and transmission rate), environment (for example, scalability, and availability), and security (for example, security and privacy). MCC is regarded

as a result of the overlap of powerful yet affordable mobile devices and Cloud computing. Mobile devices have been studied to form a virtual cloud computing provider that would benefit mobile users for their computing-intensive tasks (Huerta-Canepa & Lee 2010). Talia and Trunfio (2008, 2010) proposed a Service-Oriented Architectures for mobile DM. An overview of MCC including the definition, architecture, and applications have been surveyed in (Dinh, Lee, Niyato & Wang 2013). The issues, existing solutions, and approaches to MCC were presented in their work. Existing mobile cloud computing applications, and speculated future generation mobile cloud computing applications, were surveyed in (Wang, Chen & Wang 2015). This survey studied the existing technologies and challenges in mobile computing and MCC, which provided insights for building the next generation of mobile cloud applications. They provided existing solutions, identified research gaps, and suggested future research areas for each of the challenges.

Research on constructing networks over a number of mobile devices, which are called *Distributed Mobile Data Mining (DMDM)*, has been attracting widespread and intense interest in the last few years. DMDM aims to extract interesting patterns from mobile devices and to provide support for decision makers to make decisions relating to mobile users in a mobile environment.

DMDM is commonly used in three scenarios (Talia & Trunfio 2010);

Scenario (1) Mobile devices are terminal devices for ubiquitous access to a remote server, where DM services are provided. In this scenario, the server analyses the data stored in a local or distributed data set, and sends the results generated from the DM services to the mobile devices for their visualisation.

Scenario (2) The mobile devices gather the stream data generated in themselves and send to a remote server to be stored into a local data set. Data are periodically analysed in the server by using specific DM algorithms and the results can be used for making decisions for a given purpose.

Scenario (3) The mobile devices are used to implement the computing tasks for DM services.

Although significant advances have been made in DMDM, issues that DMDM is still facing include the typical issues in DDM environments, plus additional technological constraints such as lower bandwidth networks, slower processors, limited storage space, short-lasting battery power, and small screens to visualise the results (Pittie, Kargupta & Park 2003). Meanwhile, it is still not realistic in most scenarios to perform an entire DM task on a single mobile device due to its limited computing capability and storage space. However, a single mobile device has the capability to undertake some subtasks of a DM task (i.e, data selection and preprocessing) (Talia & Trunfio 2010).

Wang et al. (2003, 2005) proposed a distributed mobile algorithm for global association rule mining over a relational data set. Instead of shipping all local data to one site, their algorithm counts all itemsets in each transaction by growing a prefix tree when the algorithm first scans all transactions. The local and global support count could be calculated by only manipulating the tree. However, performance is still their major concern. They proposed an algorithm, Distributed ScanOnce Algorithm, designed for a distributed system to compare with a classic association rule mining algorithm, Apriori. In this algorithm, a prefix tree for each transaction is constructed and all subsets of the transaction itemset are enumerated after comprehensively considering the contribution from each transaction. According to the algorithm, the previously-scanned transactions are not stored and re-scanned and are discarded after a single pass.

One common weakness of all previous DMDM research has been pointed out by Goh et al (2005) that all transactions are gathered from a group of mobile devices and then sent to a destination, without being analysed or modified . However, many of these transactions are irrelevant, repetitive or even contain corrupted data. To solve this issue and to accelerate the MDM process, an efficient model aiming to do minor analysis of data was presented in their work. The proposed model performs minor data analysis and generates a summary of the data before sending the original data to the DM machine. When the original data arrive to the DM machine, they will be analysed in the form of summary transactions, which reduces the amount of further processing required by the DM task.

Mining time-critical financial data from a hand-held Pocket Digital Assistant (PDA),

another MDM system, called MobiMine, has been presented in (Kargupta, Park, Pittie, Liu, Kushraj & Sarkar 2002). The MobiMine adopts client-server model, which is designed for currently available low-bandwidth wireless connections between the clients and the server. MobiMine applies several advanced DM techniques and offers a variety of different tools to monitor the stock market at any time from any location. The MobiMine does not initiate an action, but is triggered by some activities in the market. In summary, " the MobiMine system is designed based on a minimalist principle: *anything that can wait should wait and therefore the MobiMine does not need to support it*".

The VEhicle Data Stream mining (VEDAS) system (Kargupta, Puttagunta, Klein & Sarkar 2006) is another example of MDM, which monitors and mines vehicle data streams in real-time in a mobile environment. The system is designed to monitor vehicles using onboard PDA-based systems connected through wireless networks. VEDAS concurrently analyses the real-time " data generated by the sensors located on vehicles, identifies the emerging patterns, and reports them to a remote control center over a low bandwidth wireless connection ".

Applying data stream classification techniques in mobile ad hoc distributed environments has been studied (Stahl, Gaber, Aldridge, May, Liu, Bramer & Yu 2012). Open Mobile Miner tool (Haghighi, Krishnaswamy, Zaslavsky, Gaber, Sinha & Gillick 2013), MobiMine (Kargupta et al. 2002) and, VEDAS (Kargupta et al. 2006) allow DM algorithms for data streams to be implemented on smartphones or tablet computers.

A DMDM framework on small devices through web services in the first scenario (in Section 2.7.2) is presented (Talia & Trunfio 2010).

There are a number of applications of using different DDM techniques. For example, mobile DM and multi-agent DM are combined to lead in a new direction, *mobile agent DM*, which can solve more pervasive problems in the world. *Mobile computing agents* have been originally proposed and discussed in (Chess, Harrison & Kershenbaum 1996). Research on them forms one of the most popular areas in DDM. Research in this area includes applications on database improvement (Indrawan, Krishnaswamy & Ranjan 2003), E-business (Ariwa, Senousy & Gaber 2003) and medical

diagnosis support (Mateo, Cervantes, Yang & Lee 2007). However, its system and the applications on its system have not been studied widely and thoroughly (Yang, Honavar, Miller & J.Wong 1998).

The fact that cannot be ignored is that applying a group of mobile devices, instead of powerful computers, as a solution for data processing has not been widely accepted by academic and industrial areas. There are a few significant issues to restrict the acceptance. The most significant are their unstable communication over wireless networks, its limited storage and computing capability, and lack of efficient management approach or system for participating devices. Additionally, a critical issue is its energy. Reducing the energy consumption in mobile devices is always an issue when mobile devices are involved in processing a task collaboratively (Rahman, Gao & Tsai 2013). However, this point only becomes important when charging is not convenient or applicable to the participating devices. Most research utilising a number of mobile devices to process data is conducted under an assumption that connections between the participating components are stable, for example, assuming the connection to the Cloud is stable (Chun & Maniatis 2009). However, in reality this assumption may not always be true. Some research adopts back-up strategies to minimise information loss because of unstable connections (Vallina-Rodriguez & Crowcroft 2011). An ad hoc mobile Cloud framework was proposed to delegate the majority of the task to nearby mobile devices that are running the same task (Huerta-Canepa & Lee 2010). To save energy in mobile devices, in an ad hoc or Wi-Fi situation, resource management in mobile devices and task delegation in ad hoc mobile Cloud could be considered.

Although these studies contributed a great deal towards improving performance when a network of mobile devices is adopted to process data, performance has been improved for a data processing task over a network of mobile devices without connections to an internet or computing Cloud.

Moreover, currently, much data processing research into using a network of mobile devices concentrates on exchanging or processing small volumes of data. However, when the volume of data is not small, either computing on single mobile device or exchanging information among mobile devices within a network of mobile devices,

there is a lack of efficient performance due to the limited computing resources of an individual mobile device and the unstable and lower bandwidth of the network (Lee, Choi, Lim, Suh, Gil & Yu 2010). These challenges hinder a network of mobile devices processing medium-large volumes of data when applicable in industrial or commercial areas.

2.8 Summary of Part II

When computing-intensive tasks are processed, computing-intensive data have not been clearly defined until the term, Big Data, was widely and seriously discussed in the mid-1990s. It has been accepted by academic and commercial fields that processing Big Data is computing-intensive. However, when a set of data is not big enough to be regarded as Big Data, whether it can still be regarded as computing-intensive, and whether it could be processed by the strategies that are used for Big Data are still two remaining questions.

To investigate the answers to these two questions, Part II first reviews the existing studies related to the definition of *computing-intensive* data. Second, DM, a common approach for processing Big Data, was reviewed. DM has experienced many contributions on theory, application and product in academic, industrial and commercial fields. However, most algorithms are complicated and need to occupy computing resources for installation and implementation. There are two major questions;

(1) whether these DM algorithms, knowledge and systems, especially working in a distributed manner, could process *non*-Big Data as well as computing-intensive data; and

(2) what modifications or changes these existing PDDM algorithms need to make to be compatible for organisations with low budgets, as well as being able to use the processing system at any time (flexibility).

There is also research regarding grouping mobile devices to provide a more flexible and open approach for computing-intensive data processing tasks for example, DDM. However, mobile devices in DDM processes are mostly data collectors, and rarely

as computing devices. Several researchers mentioned the low performance of mobile devices as the major issue. In Part III, the review focuses on what issues in performance hinder mobile devices from participating in computing-intensive data processing as computing devices.

Part III Computing-Intensive Data Processing over Wireless Networks with Mobile Devices

The low performance of mobile devices within a computing-intensive data processing task is regarded as an obstacle that prevents mobile devices from participating as computing devices. Part III reviews the procedures of computing-intensive data processing in order to see what factors affect the performance. As a well known computing-intensive data processing strategy, DM is taken as an example strategy for reviewing the procedure.

Performance of a data processing activity could be affected by two major perspectives, each processing a number of factors depending on the scenarios. The two major perspectives are: the infrastructure of the devices, and the algorithms/strategies applied on the devices (Nambi, Vasirani, Prasad & Aberer 2014). To evaluate the performance of proposed architectures, Nambi et al. (2014) modeled the indicators for the significant cost such as energy consumption, processing power, storage and communication requirements.

Performance becomes especially important for a large data set processing activity over a network of mobile devices due to a number of reasons.

- The mobile network is normally unsteady; the mobiles within it can lose connections to the network at any time. The more time the activity performs over the network, the more possibility some mobile devices lose their connection to the network, which can cause high latency or even lose data being transferred over the network;
- Mobile devices in the network have limited computing resources, such as RAM, cache and battery. If the performance of a processing activity is low, resource consumption overhead in these mobile devices could even violate their resource constraints. Therefore, computing resource management for mobile devices and scheduling strategy for a process are important.

Although processing computing-intensive tasks through a network comprising mobile devices has been attracting lots of research interest, study on how to improve the performance of an overall processing task is still lacking. If the performance of data processing over mobile devices networks cannot be improved, there is no benefit for applying mobile device networks instead of traditional distributed networks. Meanwhile, in a number of specific scenarios where traditional distributed wireless networks are not applicable, self-formed networks constructed by mobile devices are essential for computing tasks. Accordingly, improving the performance of this self-formed mobile network is important for completing computing tasks.

2.9 Factors Affecting the Performance

The rest of the section is organised by reviewing the factors affecting the performance of computing-intensive data set processes from each step of the processes.

A typical procedure of a DDM approach includes two major phases (depicted in Figure 2.2);

Phase (1) Applying a traditional centralised DM technique to the partitioned data which is distributed on different sites; and

Phase (2) Combining the results from all data sites with minimum data transmission between data sites.

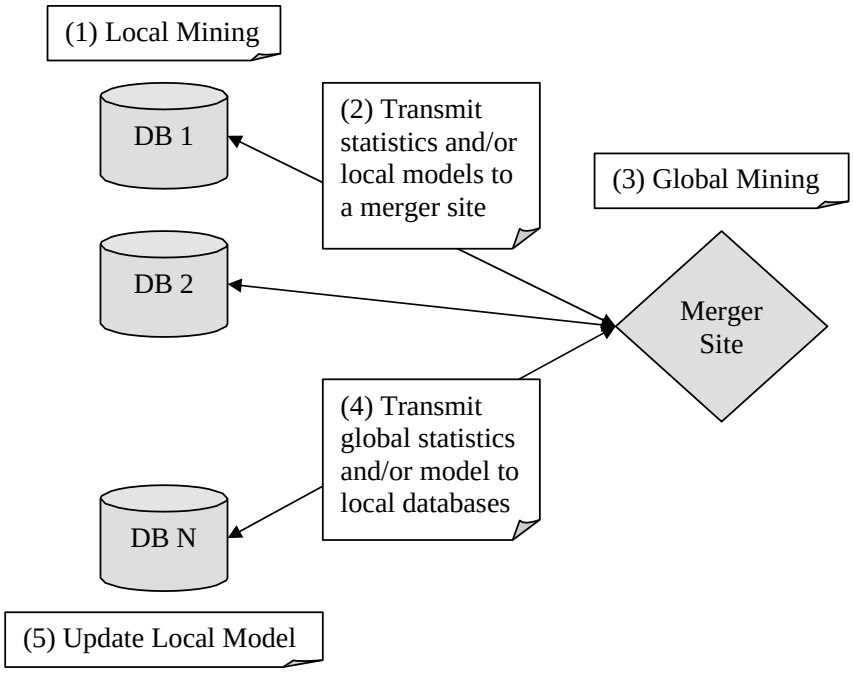

Figure 2.2: Typical procedure of a distributed data mining approach (Tsoumakas & Vlahavas 2008)

In the first phase, the local databases at each distributed site are normally analysed. The second phase involves the transmission of the discovered knowledge to a merger site, where the distributed local models are integrated in the third phase. In the fourth phase, the results are then transmitted back to the distributed databases, so that all sites become updated with the global knowledge as shown in the last phase. In some

approaches, instead of adopting a merger site to get the global model, the local models are broadcasted to all other sites, each of which computes the global model in parallel.

The architecture has been adopted by a number of research (Park & Kargupta 2003, Silva, Giannella, Bhargava, Kargupta & Klusch 2005, Kriegel, Kroger, Pryakhin & Schubert 2005, Bhamra, Verma & Patel 2015) to process a distributed data set; firstly, the same algorithm functions produce local results or models on each site of distributed data; subsequently, all local results/models are combined to produce a final model .

If the step of a data preparation is included into a distributed data processing procedure, then the procedure could be extended and detailed by four steps: data preparation (including sensitive data preservation), data set distribution, data processing and result aggregation/fusion.

Many research realised the performance of DDM has been affected by the time-consuming data transmission over network. Furthermore, when the networks that connected computing nodes are wireless networks, the performance is further affected.

Within a distributed wireless multi-sensor environment, the objective of an approach is to determine the optimum local and global decision rules that maximise the probability of signal detection. The data fusion approach used in this solution is that all local decisions, which are made by each sensor, are then combined at a fusion centre to produce a global decision (Wu, Rao, Barhen, Iyengar, Vaishnavi, Qi & Chakrabarty 2004). Typically the decision requires hypothesis testing techniques. The Bayesian and the Neyman-Pearson criteria are often used for this purpose (Park & Kargupta 2003). The other work for data fusion in Distributed Sensor Networks can be found in (Wu et al. 2004).

To generate a global mining result for distributed heterogenous data sets, a Collective Hierarchical Clustering (CHC) algorithm transmits the local dendrograms, which have been generated by applying a chosen hierarchical clustering algorithm to a local dataset, to a facilitator site; then using a statistic tools generate the global dendrogram (Johnson & Kargupta 1999).

When data are stored and managed in several distributed sites, distributed DM techniques are required to find global models representing the complete information. How-

ever, transmitting all local data sets to a DM center is often unacceptable due to privacy and security concerns, performance considerations, and bandwidth constraints (Kriegel et al. 2005). These issues also happen in the environment that mobile devices are involved in distributed DM process. In order to improve the performance of the process, Kriegel et al. (2005) proposed a distributed model-based clustering algorithm. This algorithm uses EM for detecting local models in terms of mixtures of Gaussian distributions. Their algorithm efficiently and effectively derives and merges the local Gaussian distributions to generate meaningful global models.

Traditional DM approaches for mining frequent itemset are typically based on the assumptions that data are centralised and static. When data are distributed, such approaches impose excessive communication overhead and when data are dynamic, they waste computing resources. One research attempted to overcomes these assumptions by proposing an unified approach. This approach utilises parallel and incremental techniques to generate frequent itemsets in the presence of data updates without scanning the entire database, and has minimum communication overhead when mining distributed databases. Additionally, local and global frequent itemsets are able to be generated by using this approach. This ability permits this approach to identify high-contrast frequent itemsets, which assists users to discover how the data are skewed over different sites (Otey, Wang, Parthasarathy, Veloso & Meira 2003).

Association Rules Mining (ARM) (Agrawal, Imieliński & Swami 1993, Agrawal & Srikant 1994) is one of the most popular DM techniques, which is the automatic discovery of pairs of itemsets that frequently appear together in a given large database. ARM is often a inherently disk I/O and computing and time intensive process because of the enormous database scans. The reason for scanning the database a number of times is the number of possible itemsets to be examined for *support* is too large to be done in a single scan of the large database. These issues degrades the performance of processing due to a high number of *data set scans*, *multiple phases of synchronizations* and *communication*. Belbachirs et al. (2012) pointed out that in order to reduce the I/O costs in distributed ARM, two approaches are normally applied: reducing the times for database scanning and paralleling the mining task or data set. They also presented a parallel version of the Partition algorithm (Savasere, Omiecinski & Navathe 1995).

This Partition approach divides the database into several horizontal partitions and each partition is executed independently. In the first scan of the database, a set of frequent local itemsets for each partition is produced. The global candidate itemsets are generated as being the union of all frequent local itemsets from each partition. To obtain the total *support* for these itemsets, a second scan of the database is necessary. Although the algorithm is called *Partition*, its entire computation is completed within one machine. In order to reduce the database scan, Belbachirs et al. (Belbachir & Belbachir 2012) described the parallel approach for *Partition* algorithm. Their parallel approach adopts a set of N sites (clients) managed by a coordinator site (server) , which makes the union of all local frequent itemsets and identifies the global candidate itemsets.

Distributed Association Rule Mining (DARM) is the mechanism for generating the globally strong association rules from the global frequent itemsets in distributed environments. The intelligent agent-based model for generating association rules is characterised by coordinating a variety of agents to communicate and cooperate with each other to find the global association rules in data sets. The model is a popular approach to constructing DDM systems when used to mining algorithms are required to scale up in order to process large volumes of distributed data (Bhamra et al. 2015).

The issues of mining association rules or frequent itemsets in a distributed manner have also been discussed in (Otey et al. 2003, Ashrafi, David & Smith 2004, Belbachir & Belbachir 2012, Bhamra et al. 2015). I/O cost within and among sites are regarded as the major reason to degrade the performance of distributed ARM.

Normally, in a distributed data processing architecture, data transmission over network is more time-consuming among components than in a centralised DM architecture (Ashrafi et al. 2004). Data set distribution and result aggregation/fusion are the two major steps that needs a large amount of data transmission over networks. Consequently, the efficiency of these two steps will significantly affect the performance of the processing task. Moreover, local results fusion is the step that affects the performance more than the data set distribution as it is an essential step for any distributed DM environment while the data distribution is not. For example, when data are col-

lected through distributed sensors, data are processed locally then subresults or local models generated from the sensors are fused for final result generation. In this example, there is no need to split a data set as the data set is generated in distributed sensors.

Data set split strategy and result fusion strategy in an algorithm can also highly affect the performance when mobile devices are involved in a DDM task due to the unstable and inefficient communication between devices.

Normal strategies for distributing data sets and fusing results could also be applied in a non-mobile environment. Data distribution strategies are normally mentioned in the resource allocation or scheduling phase in DDM and PDM procedure. An approach to distributing the data set and aggregating/fusing the local models/subresults for distributed density based clustering on a grid platform has been proposed in (Le-Khac, Aouad & Kechadi 2007). This approach is based on two major technologies: the extension of local models created by traditional DBSCAN algorithm at each site of the system and the aggregation of these local models by using tree-based topologies to construct global models.

To gain computation and I/O bandwidth, Arlia and Coppola (2001) proposed an approach which mapped the overall structure of DBSCAN (a clustering algorithm in DM) to a skeleton-structured (R*-tree) program that performs parallel exploration of each cluster . In their approach, the data set does not have to be split up. The data set is still centralised, but it could be processed in a parallel manner to enhance the performance of region queries which account for 95% of the computation time even when the R*-tree fits in memory and contains the data with only two dimensions. Therefore their research proposed a parallel version of the ExpandCluster algorithm by using the Master and Slaves mode. The Master module performs cluster assignment, while the Slave module answers neighborhood queries via R*-Tree.

Mobile devices possess the limited and highly dynamic computing resources, such as available processor time, available memory and network quality. When they are involved into a computing-intensive task, task and resource allocation approaches within its data processing strategy become more critical and must be seriously taken into ac-

count. Comito et al. (2011, 2013*a*, 2013*b*) studied the allocation approaches in mobile environment and proposed a clustering scheme using multiple criteria to optimise the mobility similarity of the devices within a group and maximise the lifetime of a network (2014).

Another factor to affect the performance of a data processing system is the control measures that the system adopted to deal with users' privacy concerns. If the data owners are *unwilling* to share their data because of privacy issues for a collaborative data processing task, the performance of the data processing task will be affected. For DM purposes, instead of providing their raw data, data owners are encouraged to provide statistics features about their data or their encrypted data under specially designed cryptographic protocols (Aggarwal & Yu 2008).

Identifying high risk sensitive data is a significant task in the field of *risk management*. When considering how privacy is to be protected in enterprise, because of financial concerns, most companies/institutes spend their major efforts on the most sensitive information, i.e. the information in the highest risk. To do this, companies/institutes must understand the relative value of different information by providing solutions of identifying high-risk data through a number of consultant services (Price Waterhouse Coopers 2011).

Huang et al. (2010) discussed the security issues identified in MANETs, including;

(1). The security of existing MANET infrastructure lacks interoperability support in a heterogeneous communication environment. Communication devices that belong to different domains and are equipped with different computation and communication capabilities make the design of the protocol in security extremely difficult. These difficulties are commonly caused by a large number of uncertainties when communications peers attempt to setup the security protocols based on their agreements. For example, mobile users may use different identity spaces, cryptographic parameters, and reside in different administrative domains;

(2). The mobility feature of MANETs imposes a significant impact on the security and communication performance in location tracking, communication privacy, reliability and survivability. Uncertainties introduced by the mobility produces

unpredictable delays or even failures in transmission. Thus, the MANETs require a comprehensive approach to conducting risk assessment under the considerations of security and communication requirements.

With the increasing use of mobile devices in Mobile Cloud Computing (MCC), the study of performance in MCC has been attracting lots of attentions. In MCC, mobile devices do not need to have large storage capacities and powerful processors. Data are stored or processed in the Clouds with various mechanism, where data security becomes the major issue (Garg & Sharma 2013). Because of this issue, the performance of processing tasks in MCC has been highly affected, which makes many IT professionals not showing interests towards MCC. As a result, research needs to be carried out on overcoming the security issues and finding solutions to provide secure access control to the data, which will assist to improve the performance of tasks if the solutions are efficient and convenient.

Another area is to apply Cloud computing to processing data sets from ad hoc mo-bile devices. They either send the distributed data set in mobile devices to a Cloud for processing (Gaber et al. 2014) or they temporarily create a task-based or member-based virtual Cloud (Huerta-Canepa & Lee 2010). These Cloud-based solutions group a number of mobile devices to undertake middle-large data processing tasks. As part of the research in the *FocusDrive* project, which aims to improve teenagers' driving safety, dynamic location information from mobile users is defined as sensitive infor-mation in (Huang, Xu, Xing & Zhong 2011). Also, this research presented a solution using Extended Semi-Shadow Images (ESSI) to select an anonymous identity with a certificate issued by the Trusted Authority (TA) to authenticate the trace data. In a Mo-bileCloud framework, a mobile device outsources its computing and storage services to its corresponding ESSI and Secure Storage.

Since, privacy issues have been mentioned and discussed in DM over sensor networks by research including a survey (Chan & Perrig 2003) and privacy invasions exploiting transmission pattern analysis and statistical inference in the Smart Home environment (Park, Basaran, Park & Son 2014). Another application of sensor-based networks are healthcare and medical science in which privacy concerns are significant (Agrawal

2015).

2.10 Summary of Part III

One of the major issues when processing computing-intensive data by a network of mobile devices is low *performance*. Low performance of DDM procedures caused by the involvement of mobile devices is the major reason that hinders mobile devices from being the participants in the procedures of computing-intensive data processing.

The performance has been affected by a number of factors, such as limited storage, limited computing capability and small energy capacity. These factors regarding the nature of mobile devices have attracted more research interest.

However, I realised that a computing-intensive data processing procedure is comprised of many steps or components, mobile devices could not be the dominating reason that affects performance. Factors from each step of the procedure should be also investigated from the aspect of overarching organisation.

Unfortunately, research on how each step of an entire processing procedure affects performance has not been found, especially when the network of mobile devices has no connection to an internet or a Cloud service.

Part III reviews the procedures of DDM in order to investigate the factors that may affect performance.

Factors affecting the performance of data set processing activity over a network of mobile devices are considered from the different steps of the activity, data distribution, data processing, result fusion, and sensitive data preservation if the data owner has privacy concerns.

Part IV Summary

With the prevalence of mobile devices, recently they are considered as being able to participate in the computing networks. Although the computing capability of an individual mobile device may not be powerful, a number of these devices can be grouped or networked as a computing service provider with the following advantages (Huerta-Canepa & Lee 2010);

(i) the mobile devices are becoming more common, which increases the possibility of form the available nearby devices as a computing network;

(ii) over time the mobile devices are being equipped with more powerful computing capabilities;

(iii) the self-formed network allows devices to communicate with each other with no further financial cost; and

(iv) the devices allow us to create our own communities in which shared tasks can be executed frequently with a minimum privacy concern.

Nowadays, processing data by using a number of small computing devices commonly needs an internet for data transmission or the computing Cloud for distributed computing or data storage. However, in some situations where are no connection to an internet or a computing Cloud, when a data owner needs to process data and data owners have no capability for processing the data by using their own devices, a local self-constructed network is considered as a solution. In this situation, a network of mobile devices could become the solution for processing a computing-intensive task. Especially when a computing task could be done over a network of mobile devices within several hours, compared to a half day by a single desktop and considering the battery life for some mobile devices, processing data within a network of mobile devices is worthwhile studying.

This chapter reviews basic concepts of networks, with more focus on wireless networks and data processing strategies over wireless networks, with DM as an example of data

processing strategy. These reviews give an overview of the research areas which are covered by two major perspectives: data processing and network construction.

From the perspective of data processing, two approaches are commonly adopted to process computing-intensive data: the centralised approach and the distributed approach.

From the point of view of network construction, recent use of mobile devices to construct networks for processing computing tasks has attracted continuous research interest. However, the fact that cannot be ignored is that applying a group of mobile devices as a solution for data processing is still not widely accepted in academia and industry. The top issues are: unstable communication over wireless networks, limited storage and computing capability, and lack of efficient management approaches or systems for participating devices. Additionally, another critical issue is the limited energy of mobile devices when a processing task is energy-consuming. In summary, all these issues affect the performance of computing-intensive data processing tasks. A lot of research has contributed to improve the performance of such tasks. However, little research has been found on how to improve the performance of overall processing tasks.

Moreover, most research on processing data with a network of mobile devices deals with small volumes of data. When the volume of data is not small, there is a lack of applications of a network of mobile devices to process these data.

There is yet a solution to be found in the past research to those scenarios in which a network of mobile devices is the only option to process medium-large data. Also, research is lacking on how an entire processing procedure affects performance, especially when the network of mobile devices has no connection to an internet or a Cloud service. After reviewing the common DDM approaches over wireless networks, apart from the battery restriction, I realise that other factors that prevent mobile devices from participating as computing devices may be investigated from the perspective of each *single* step in a data processing procedure.

These findings from the literature review motivate the research interest of this thesis: *applying intensive data processing over a group of mobile devices with less powerful computing or processing capability from the perspective of the entire processing procedure*.

This interest guides this research to explore strategies for improving the performance at each stage of implementing a computing-intensive data processing task over a network of mobile devices. These strategies includes data partitioning, sensitive data protection, data processing and result aggregation, and includes the strategies to predict:

a group of the most suitable participating mobile devices (details are presented in Chapter 3);

the most efficient processing algorithm suitable for a processing task (details are presented in Chapter 4); and

the most appropriate amount of sensitive information in a data set (details are presented in Chapter 7).

Additionally, Chapter 5 demonstrates a series of experiments on the strategies proposed in Chapter 3. Chapter 6 proposes a framework to support the procedure of data processing, followed by a case study of this framework in order to demonstrate the utility of the performance improvement strategies. Chapter 8 draws a conclusion and presents future work.

Chapter 3

Prediction Strategy for Improving Performance over a Network of Cooperative Flexible-sized Mobile Devices (FlexMNet)

In terms of the definition of Big Data, this chapter firstly defines *medium data*, which can be processed by a group of current mobile devices in a reasonable time.

Medium data, data transmission and communication between mobile devices become the major factors that affect the performance. Therefore, performance can be improved by reducing the possibility of reallocating a data set to other mobile devices and of re-balancing the computing workload in each device. To achieve this goal, this chapter presents a prediction strategy of the optimal group of mobile devices for a processing algorithm and the dataset. This prediction strategy is conducted before the actual data processing task is running.

3.1 Medium Data

Data are considered to be *Big Data* when the volume is over one terabyte according to the survey conducted by IBM in mid-2012 (Schroeck, Shockley, Smart, Romero-Morales & Tufano 2012). In 2016, most mobile devices, e.g. laptops, wearable devices and smartphones could store one terabyte of data in their memory. One terabyte of data often needs more than one day of processing time by an up-to-date personal computer, although the actual processing time varies depending on the type of processing. Therefore, this research takes the definition of *Big Data* and *Medium Data* from the perspective of computing overheads, combining the two dimensions Volume and Variety, which have been discussed in (Laney 2001, Chen et al. 2012, O. Kwon 2014):

> *Big Data* in this research is regarded as a data set which needs to be processed by a single up-to-date personal computer for more than a day, regardless of how powerful a computer is, or the data structure and processing algorithm it uses.

According to the definition of *Big Data* above, *Medium Data* is defined as

> the data set which generally needs an up-to-date personal computer, applying an existing centralised data processing algorithm to process data for more than half a day but less than one day.

Within the scenario that this research introduced of using a network of MDevs, *Medium Data* can also be simply understood as *ten* GigaBytes (GBs) to *one* TeraByte (TBs) of storage compared to the average storage space in an up-to-date smartphone or an up-to-date laptop. The smartphone could be equipped with twenty two to sixty-four GigaBytes of memory space while a laptop could be equipped with one TeraByte memory space, as has been the case since about 2015.

Giving accurate quantified definitions of both *Big Data* and *Medium Data* is impossible due to the fact that quantifying data is varied by a number of factors. For example, the year that the definition is taken from (one GB was regarded as *Big Data* in 2000, while this will not be the case in 2020); the performance of computing devices; the data structure; the algorithms used to process the data; and the understanding of the descriptions *Big* and *Medium* by the person/organisation who gives the definitions.

Although *Medium Data* is not computing-intensive compared to *Big Data*, it can not be processed by data owners themselves, and instead needs assistance from other computing devices in real world cases. For example, the computing capability of data owners cannot satisfy the computing overhead of the data set.

A medium volume of data is applied in a working scenario to present the issues of data processing over a network of mobile devices and to demonstrate the work of this research.

The next section describes an example scenario of a network setup in the real world, where a set of medium data needs to be processed collaboratively by a group of mobile devices.

3.2 Working Scenario

When a number of flexible-sized mobile devices are connected by a wireless network or several wireless networks and dedicated to collaboratively implementing certain application(s) under a series of agreements, these mobile devices form a Flexible-sized Mobile Devices Network (FlexMNet). One of the applications of FlexMNet is to process medium data. The following scenario gives an example of utilising FlexMNet.

This example scenario is used for the proposed strategies throughout this thesis.

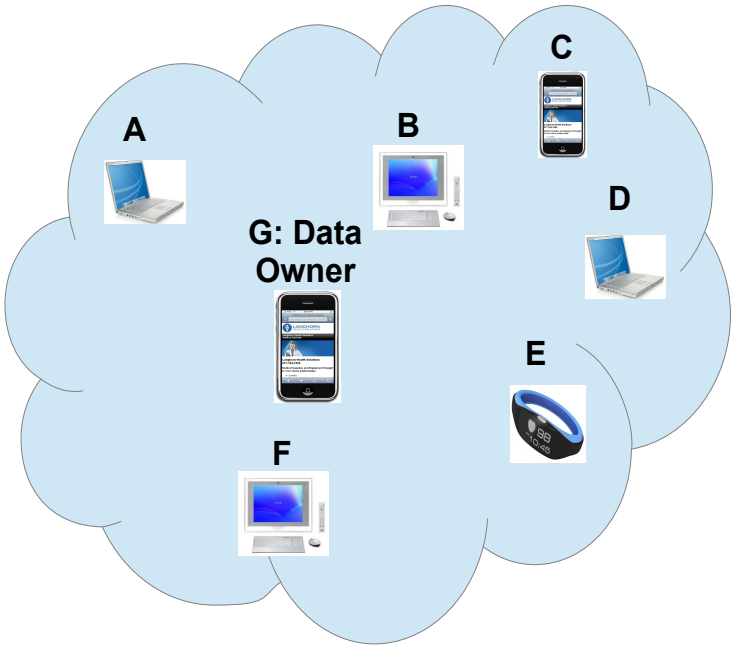

Figure 3.1. An example of FlexMNet

A number of mobile devices, illustrated as (A,B,C,D,E,F and G) in Figure 3.1, are connected by a wireless local network created by a hospital. Those mobile devices are owned by the hospital and provided by the hospital to its staff members for work purposes. Due to security reasons, these mobile devices are only allowed to connect to the local network and the network does not connect to an internet service, e.g. the Internet or a Cloud service.

One of the staff members, who owns the mobile device G, has a request to process the data stored in his/her mobile device. This staff member is called a data owner (DO) in the hospital premises. The data is Medium Data and the DO does not have the computing capability, for example, power or memory, to process the data themselves. In order to process the data, the DO needs computing assistance from other mobile devices (A, B, C, D, E, F) which are connected by a wireless local network and are available and willing to contribute their computing resources to collaboratively assist the DO. Those mobile devices (A, B, C, D, E, F) do not have

an equivalent computing competency.

This research studies approaches to improving the performance of processing medium data over a network of mobile devices. This thesis focuses on a centralised data set, such as pictures, videos and alphanumeric data sets. Data streams are out of the scope of this research. To improve performance, which mobile devices, from those available (for example, A-F in the scenario described above), would be involved in processing the data, of which the size and structure are known?

As summarised in the Chapter 2, communication between mobile devices is one of the major factors affecting the performance of a computing-intensive task over a network of mobile devices. Consequently, *reducing transmission and communication* becomes an important task for improving performance. Once an algorithm to process the data set has been chosen, the manner of the communications between computing devices (for example, the timing, frequency and types of exchange message) is also determined. It is beneficial for improving the performance of a data processing task if the data is partitioned proportionally to the computing capabilities of each computing device. The reasons are twofold;

(i) If its processing algorithm needs devices to exchange intermediate results a couple of times, then each mobile device would generate intermediate results at a similar time, which would not cause a situation where one device generating the result earlier has to wait for another one to generate a result.

(ii) Under the condition that its processing algorithm needs only two transmissions; one is allocating the original large data set into devices; the second is that devices send results back to the DO. In each device, its computing capability just matches the computing capability required by the allocated data set.

Meanwhile, proportionally partitioning the data set in terms of the computing capabilities of the computing devices can avoid the following scenarios which can degrade the performance;

(i) Some devices have been allocated a data set which is beyond their computing capability, resulting in a failure to complete the task;

(ii) Some devices have not been allocated a sufficient data set, resulting in the wasting of their computing resources;

(iii) Data packet loss or data transmission failure is caused by the instability or the bad quality of the network.

All of these reasons would delay the completion of the computing task. It is important to study a strategy that predicts which mobile devices are suitable to collaboratively undertaking the computing tasks with the best performance and least processing time. This research proposes a prediction strategy to improve the performance of *Medium Data* over a network of mobile devices.

3.3 Terminology

3.3.1 Mobile Device

Mobile Device (MDev) is defined in this research as any digital device which has the following features:

- Computing capability;
- Storage capacity;
- Being able to be moved to any other locations;
- Being able to connect to wireless networks.

In terms of the above features, it can be a smartphone, a tablet, a laptop, a wearable digital device or a desktop personal computer which is not connected by a wired network.

Computational resources of a MDev are used to undertake a computing task. They are memory space (RAM and secondary memory) for data storage, algorithms to process data and processors to execute the algorithms on data.

3.3.2 Computing Unit

Computing Unit (CU) is a virtual computing component which represents a certain amount of user-defined or system-defined computing competency. In the proposed approach, it is defined as the basic computing task carrier to *independently* conduct a piece of computing workload. *Independently* in this context means that it can generate an output specified by the workload facilitator from an input of a data set after being told the format of an output and without being interrupted during data processing.

Size of a CU is defined by the time that a CU spends on processing a given data set. The time comprises three components:

(1) Loading the data set into the CU;

(2) Processing the data set;

(3) Generating a desirable result.

The Size of a CU varies in terms of network conditions. Once a CU is determined, values for components (2) and (3) will not change. However, the first component, the time to load data into the CU, is changeable depending on the wireless network speed. For this reason, when the network environment changes, the size of all CUs needs to be recalculated.

3.3.3 Contributor

Contributor in this thesis is defined as *a virtual and active and team-spirited entity which has certain properties and behaviors*. If a MDev contributes its computational resources to a data processing task, it is normally called a *Contributor*. The structure of a *Contributor* is illustrated in Figure 3.3.3.

The idea of a Contributor used in this thesis originates from the concept of *Object*. In the domain of object-oriented programming, an *object* usually means an entity being encapsulated with a package of attributes (object elements) and behaviors (methods or subroutines) (Poo, Kiong & Ashok 2008). By considering the characteristics of *object*, *Contributor* is defined by four properties combined as the *Token* of a *Contributor*:

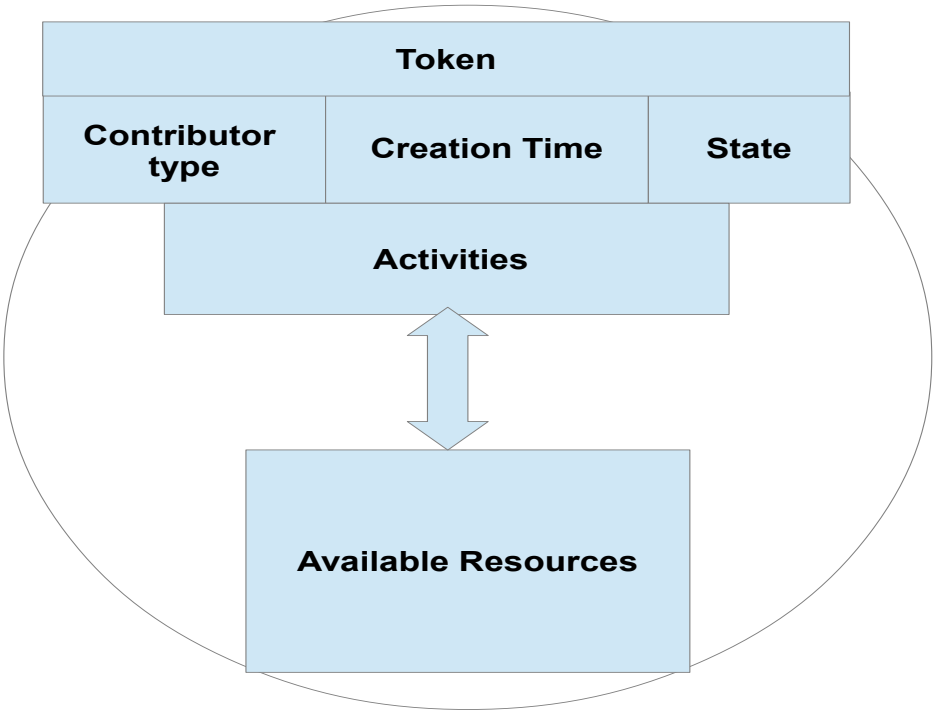

Figure 3.2. Structure of a Contributor

- *Identity* indicates the properties of a *Contributor* that distinguishes it from other *Contributors*. The properties include *Contributor type*, information about the MDevs in which *Contributor*s locate and information about resources that the Contributor would like to contribute.
- *Creation Time* records the time point that a Contributor is created.
- *State* describes the availability of a *Contributor*. It includes three values: *Ready*, *Occupied* and *Waiting*;
- *Activities* describes the methods/functions that a *Contributor* can perform.

Each type of Contributor has a different set of interfaces to define their activities.

Contribution Duration refers to the time duration that each Contributor commits its available computational resources to a computing task.

3.3.4 Computing Contributor

Computing Contributor (CC) is one type of Contributors when a MDev is able to contribute its computing capability to a computing task. The capability of a CC is defined by the number of CUs. For example, MDev **A** can contribute five CUs to a computing task while device **B** can contribute three CUs. In this case, **A** can contribute more than **B** to the computing task. The *Contributor type* in the Token of a CC is *Computing*.

Computing Capability of a CC refers to the maximum computing workload a CC is able to take. This definition borrows the concept of *computational capability* in a multi-processor system, in which *computational capability* is understood as the maximum computing power that a computing entity uses when solving one large problem with the shortest amount of time. It is estimated by selecting the number of large processors used to undertake a computational load required by a particular computing application (West 1967).

Computing Capability is quantified by the number of CUs. It is firstly measured by the minutes of time taken when processing a user-defined **Input n** on its associated processor(s). Then the minutes are converted to a number of CUs according to user-given principles.

Figure 3.3 illustrates an example of quantifying the computing capability of a CC generated by a MDev. Computing capability of the CC in the example MDev is 3 CUs.

Computing capability is used in this thesis to quantify the efficiency with which a computing device can solve a particular computing problem.

3.3.5 Medium Data Processing Algorithm

Medium Data Processing Algorithm (Med-data algorithm) refers to the algorithms which are designed to process medium data over FlexMNet.

Measurement factors are used to measure the efficiency of a computing algorithm with the size of the **input n** (McConnell 2008). The two most common *measurement factors* for algorithms in MDevs are as follows;

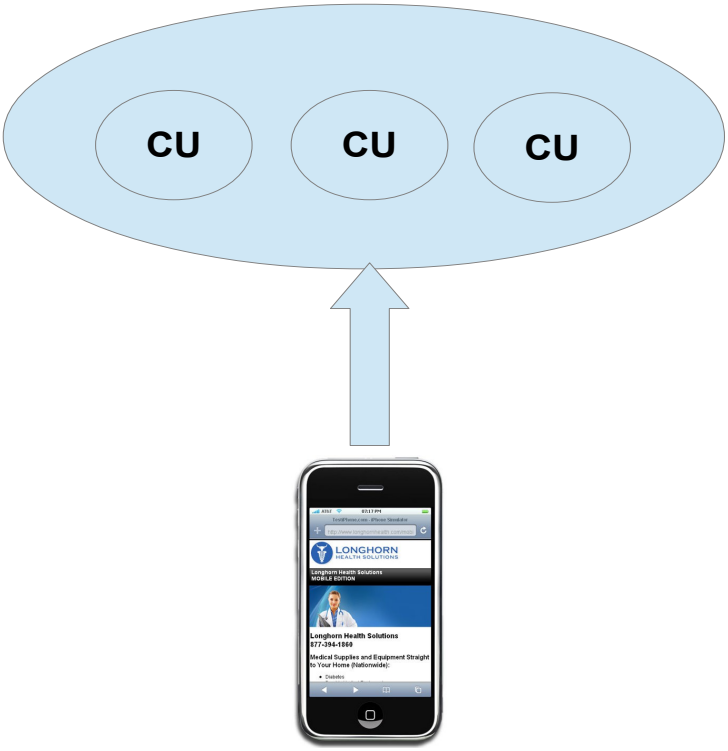

Figure 3.3. A Computing Contributor with the capability of 3 Computing Units

(1) *Processing Time* refers to the time taken by an algorithm to execute a computing task with an **input n**. It is the sum of *Transmission Time* and *Computation Time*. *Transmission Time* refers to the time taken to transmit information, e.g. data set, messages and subresults, when an algorithm is adopted to process a computing task. *Computation Time* refers to the time that a MDev takes to execute an algorithm over a data set, including time to process a data set and time to merge results transmitted from other MDevs. Therefore, *Processing Time* is represented in Equation 3.1 by the addition of time to process a data set, time to merge subresults, and time to transmit subresults. More details of calculating *Processing Time* are presented in Chapter 4.

(2) *Memory Usage* refers to the amount of working memory (typically RAM for a MDev) needed when an algorithm is executed on a data set. Memory Usage is measured by the sum of the amount of memory needed by the scripts of an algorithm and the amount of memory needed for the data and temporary results if needed, which are represented in Equation 3.2. A good algorithm keeps this Mem-

ory Usage as small as possible.

$$\text{Processing Time} = \text{Time to process a data set} + \text{Time to merge subresults} + \text{Transmission Time} \quad (3.1)$$

$$\text{Memory Usage} = \text{Memory space for data} + \text{Memory space for temporary results} \quad (3.2)$$

Apart from *Processing time* and *Memory Usage*, other measurement factors which are also important in MDev are:

- *Transmission size*: this measure can significantly affect the efficiency of an algorithm when the bandwidth of a network is a limiting factor.
- *Secondary memory*: this refers to the larger, cheaper and slower storage space which does not directly connect to processors compared to *primary memory*. It is an external memory device for a computer. It stores data files, application programs, algorithms and other resources which need long-term storage (Sipser 2010).
- *Response time*: this parameter has particularly relevance for an interactive application, where it is defined as the time elapse of responding a message or request sent from another MDev. The definition of it for MDev borrows the concept of it in operating systems (Gill 2006).

Performance Cost of a Med-data algorithm refers to time and memory cost when running a computing task over a network of mobile computing devices. The cost includes two parts: *Pure Performance Overhead* and *Extra Performance Cost*.

Pure Performance Overhead refers to time and memory cost when implementing algorithms in laboratories where traffic load over a network is constant, equipment failure and network congestion fail to happen, and network latency is predictable.

Extra Performance Cost are the time and memory cost on performance except not accounted for in *Pure Performance Overhead*. This mainly includes communication cost over network, especially when network conditions are bad. This cost is often not considered in laboratory experiments.

The communication of a MDev required by a Med-data algorithm can be classified into two categories: *Internal Device Communication* and *External Device Communication*.

- *Internal Device Communication* refers to the communications within a device where the communication does not rely on an external network. This communication is implemented as part of *Pure Performance Overhead*.
- *External Device Communication* to a MDev refers to the communication needing an external network as a transmission medium. This communication is covered by *Pure Performance Overhead*, and not*Extra Performance Cost*.

3.3.6 Mobile Group

Mobile Group (MGroup) refers to a number of MDevs within FlexMNet which are assigned to do a certain task run by a Med-data algorithm. The number of group members, 1 to N, depends on the count of group members. Group members in each group are represented by *Combination* in mathematics.

Combination in mathematics refers to an approach to selecting k items at a time without repetition from a collection with n items. A combination of n objects taken k at a time is any subset of k objects from a set of n distinct, which is sometimes called k-combination of a set S. The number of k-combination of a set S with n elements, is equal to the binomial coefficient (Vivaldi 2001):

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{n(n-1)\dots(n-k+1)}{k(k-1)\dots 1} \tag{3.3}$$

When n is fairly small, the number of combinations is countable. For example, a person needs to choose at least one MDev from 3 different MDevs (A, B and C) without repetition for an experiment. He will have the following combinations:

i) 3 combinations for 1 element ({A}, {B} and {C}); and

ii) 3 combinations for 2 elements ({A, B}, {A, C} and {B,C}); and

iii) 1 combination for 3 elements ({A,B,C}).

Therefore, in total, this person will have $3+3+1=7$ combinations.

Candidate Mobile Group (Candidate Group) refers to the group of MDevs which are selected to calculate their *Processing Time* and *Memory Usage* in *Predication Strategy* (described in later section of this chapter).

Prediction Algorithm is used to predict which available MDevs are going to be used in processing a medium data set under the condition that the algorithm used to process the data set is known.

The above-mentioned entities and their notations are summarised in Table 3.1.

Table 3.1. Summary of entities and their notations

Entities	Notations
Mobile Device	MDev
Computing Unit	CU
Contributor	Contributor
Computing Contributor	CC
Mobile Group	MGroup

3.4 Prediction for Data Set Distribution

This section describes a prediction strategy for distributing a data set to a group of mobile devices. A problem definition is given before the prediction strategy is introduced.

3.4.1 Problem Definition

N MDevs (D_1, D_2, D_3, ..., D_N) are connected by a local wireless network and are available to commit to a data processing task requested by one of

Table 3.2. All possible groups of participating devices

Set $No._{[1]}$	Description	Group No.	Group Elements	Number of Groups in Each Set
Set I	Groups with 1 MDev	Group 1	A	$\binom{4}{1} = 4$
		Group 2	B	
		Group 3	C	
		Group 4	D	
Set II	Groups with 2 MDevs	Group 5	A,B	$\binom{4}{2} = 6$
		Group 6	A,C	
		Group 7	A,D	
		Group 8	B,C	
		Group 9	B,D	
		Group 10	C,D	
Set III	Groups with 3 MDevs	Group 11	A,B,C	$\binom{4}{3} = 4$
		Group 12	A,B,D	
		Group 13	A,C,D	
		Group 14	B,C,D	
Set IV	Groups with 4 MDevs	Group 15	A,B,C,D	$\binom{4}{4} = 1$
			Total Number of Groups	15

[1]: *No.* indicates how many devices in each group.

the N devices. The device owner who requests the processing task and owns the data is a DO. $\sum_{k=N-1}^{2} \binom{N}{k}$ groups of the devices can provide their computing resources to the processing task. To optimise the performance of a data processing task, not all devices may be used for the task. A group of these devices needs to be selected to compute the task. An example is given as follows in order to demonstrate the approach to select the group.

Assuming N is 4, which means that 4 devices with different computing capabilities are available. The names of these 4 devices are A, B, C and D. Table 3.2 lists all groups comprising these 4 devices which form a computing network, available process a computing task. The number of groups is $\binom{4}{1} + \binom{4}{2} + \binom{4}{3} + \binom{4}{4} = 15$. Groups in *Set 1* does not need to be considered because the DO is looking for other devices' cooperation. Finally, only $15 - 4 = 11$ groups left in the table need to be considered. The prediction strategy proposed next describes the procedure of choosing one of the 11 groups to undertake the task.

3.4.2 Prediction Strategy

A number of groups listed in Table 3.2 are either inapplicable or less efficient to the data processing task. These groups are called *prunable groups*. A prediction strategy with a number of rules is proposed to prune these groups.

When processing a task, only the group with the best performance predicted by the prediction strategy is recommended to DOs. Thus, predicting which group may have the best performance is the aim of this strategy.

Testing processing performance for all groups will be time-consuming and also against the rule of saving energy and resources for a mobile computing task. In order to solve this issue, three steps are used to describe the strategy for minimising the number of groups for the simulation of final prediction.

Step (1) A number of preliminary rules are applied to remove the prunable groups.One example rule is that the memory requirement for the dedicated algorithm can not be satisfied;

Step (2) The DO's requirements, e.g. privacy and efficiency consideration, are considered in order to prune a number of groups;

Step (3) Selection of several groups with the least number of devices (to reduce the transmission time) and also with devices that are equipped with faster processors compared to other devices. These selected groups are *candidate groups*.

The preliminary rules applied in Step (1) include:

(a) The groups in Set I do not need to be considered because the DO is looking for other devices' cooperation;

(b) In most cases, not all devices are selected for the task due to there being more communications among devices when more devices participate. As a result, the group in *Set N* is not considered if the DO understands that the number of available devices is large;

(c) Prediction for *Computation Time* and *Memory Usage* for a computing task needs to be conducted prior to the actual implementation of a group of MDevs. For

example, there is a 1 GB data set and eight available devices. In those eight, five devices have the computing capability of processing over 500MB data set individually and another three processing 200 MB data individually. Therefore, it is not necessary to consider any groups with any of these three devices.

(d) As most MDevs are battery-driven, if they keep on executing a task for more than one hour, their performance is going to be affected because of overheating. To avoid issues of overheating or overusing a device, balancing the computing workload among devices needs to be taken into account.

After applying these preliminary rules, normally, less than $\sum_{k=N-1}^{2} \binom{N}{k}$ groups are left for calculation. For a DO the time to predict the overall performance is reduced. These groups are *candidate groups*.

In step 3, *Processing Time* and *Memory Usage* are calculated for each candidate group . The strategy for predicting the optimised group is summarised in the following pseudo code. Supposing each group has M MDevs.

In order to store the sorted result of *Processing Time* and *Available Memory*, firstly, two lists, T and S are initialised as two empty sets. List T stores the *Processing Time* with an increasing order while list S stores *Available Memory* with increasing order. Another parameter *Prediction Result* is used to store the predicted result, generated from this strategy.

From the first group to the last group, calculating *Time Taken For Sample Data* then stores the results to list T, *Processing Time*; also calculating $\sum_{n=1}^{M}$ Maximum Used Space, then stores the results to the list S, *Available Memory*.

In the Prediction Result section, if there is a group in the top three in list T and list S, then this group is chosen as the Prediction Result. Namely, the processing task will be implemented in the group of devices from the predicted result.

If there is no group in the top m in both the *Processing Time* list and the *Available Memory* list, then the user needs to give a higher priority to chose a suitable combination first from the *Processing Time* list or the *Available Memory* list. By default, the system gives a priority to *Processing Time* rather than *Available Memory* because the sooner a task is complete, the more memory blocks in devices are freed for other

input : All Candidates Group ; Parameters: list T of *Processing Time*=empty, list S of *Available Memory* =empty, *ChosenList*=empty , m: user defined number

output: Prediction Result

Section 1: Calculation

for $i \leftarrow the first group$ **to** $the last group$ **do**
  foreach *group* **do**
  Calculate *Processing Time* = TimeTakenForSampleData;
  Insert *this group* in the list T;
  Calculate *Available Memory* $= \sum_{n=1}^{M}$ Maximum Used Space;
  Insert *this group* in the list S. ;
end

Sort the two lists, T and S, and identify the top m groups in each list.

Section 2: Prediction Result Generation

foreach *group in either list T or S* **do**
  if *The group is the top m in both lists T and S* **then**
    Set this group to the Prediction Result;
    Go to Section 3;
  else if *User gives a higher priority to either Processing Time or Available Memory* **then**
      Set the list given higher priority by user as the *ChosenList*;
    else
      Set *Processing Time* list as the *ChosenList*;
    ;
  ;
  Set the *first group* in the *ChosenList* to the *Prediction Result*;
end

Section 3: Prediction Result Output

Algorithm 1: Prediction Algorithm: predicting the most efficient group of MDevs

tasks. According to the given priority, a group will be chosen from the list and given a priority. What follows is the implementation in terms of the settings in the group.

How can the intermediate results between devices be synchronised as most distributed algorithms need to exchange or synchronise some intermediate results to process the rest of the steps? In this prediction strategy, latency among devices exists, however it would not heavily affect the entire processing time for two reasons. Firstly, there is an assumption that the distribution of a data set is even, namely, the local clusters are evenly distributed. Secondly, the workload allocated into the devices is proportional to match their available processing capability. Powerful devices are allocated more workload, but they will generate the intermediate results at a similar time as the less powerful devices do.

3.5 Summary

After investigating the performance problems of a network of MDevs used to process a computing-intensive task, the frequent communication and the large data transmissions are regarded as the major issues. These issues direct this research to focus on minimising the frequency of communication and the frequency of large data transfer for a computing task in a network of MDevs. To achieve this goal, this chapter proposes a prediction strategy to estimate which MDevs can collaboratively process a known data set with a known algorithm with a higher possibility of the most efficient performance. Terminology used in the prediction strategy has also been introduced in this chapter.

My approach is novel in the context of intensive computing (defined by this thesis as *Medium Data*) over a network of MDevs, since there is no prior work that considers using a group of MDevs to process intensive computing tasks. I formally state the problem in the following section and propose a strategy to improve the performance of data analysis task over a network comprised of MDevs.

Experiments in the next chapter demonstrate the feasibility of this prediction strategy when a medium data set is processed over FlexMNet .

Chapter 4

Data Processing Algorithm Over FlexMNet

4.1 Introduction

In FlexMNet, a data set is processed by an algorithm which includes strategies for splitting the data set into several partitions and fusing local results from partitions into a global one. The algorithm is an important factor for affecting the performance of processing a medium data set over FlexMNet for several reasons.

When a data set is processed in FlexMNet, transmission between devices is built on the unstable and low speed wireless network (compared to wired networks) and each MDev has limited computing resources. Splitting a data set into a number of partitions which can be processed in each participating MDev and fusing the results from each device are two significant factors which affect the performance of the entire data processing task. If the data set has not been partitioned into an appropriate size for each device, when retransmission of the data set is needed to complete the processing, the overall performance of processing will be highly affected. Data set distribution and result fusion are two steps of distributed data processing algorithms. Additionally, an algorithm mostly determine how efficiently computing resources involved in a DDM system can cooperate. As a result, the algorithms need to be carefully studied and chosen with the efficient distribution and fusion strategies for a dedicated data processing task. Furthermore, I understand that a distributed data processing algorithm suitable for the data processing over FlexMNet should meet several requirements. For example, minimising the original data set transmission between MDevs and maximising partitions of a data set within a MDev.

The main goal of the chapter is to discuss the factors that affect the performance of processing a medium data set from the perspective of algorithms. There are several requirements that the algorithms used in FlexMNet should comply with. These includes;

i) Being light-weight, which means the complexity has been minimised in order to reduce the usage of memory space, especially in the steps of partitioning data and fusing local results;

ii) Minimising the number of data transmissions between MDevs; and

iii) Minimising the number of entire data set scans.

These factors are taken as criteria in a Ranking strategy, proposed in this chapter, in order to evaluate a number of algorithms.

Various computing-intensive data processing algorithms have their own data set split and local result fusion strategies. Association Rule Mining (ARM) (Agrawal et al. 1993, Agrawal & Srikant 1994) is a popular DM technology, which specialises in processing computing-intensive data. Therefore, this chapter will take ARM as an example data processing strategy to present the factors which affect the performance and Ranking strategy.

According to the typical procedure of a DDM approach 2.2 and the properties of ARM, a typical procedure of distributed ARM is further represented in Figure 4.1. From this procedure, communication cost over FlexMNet mainly occurs in two stages: distributing a centralised data set, and fusing local models.

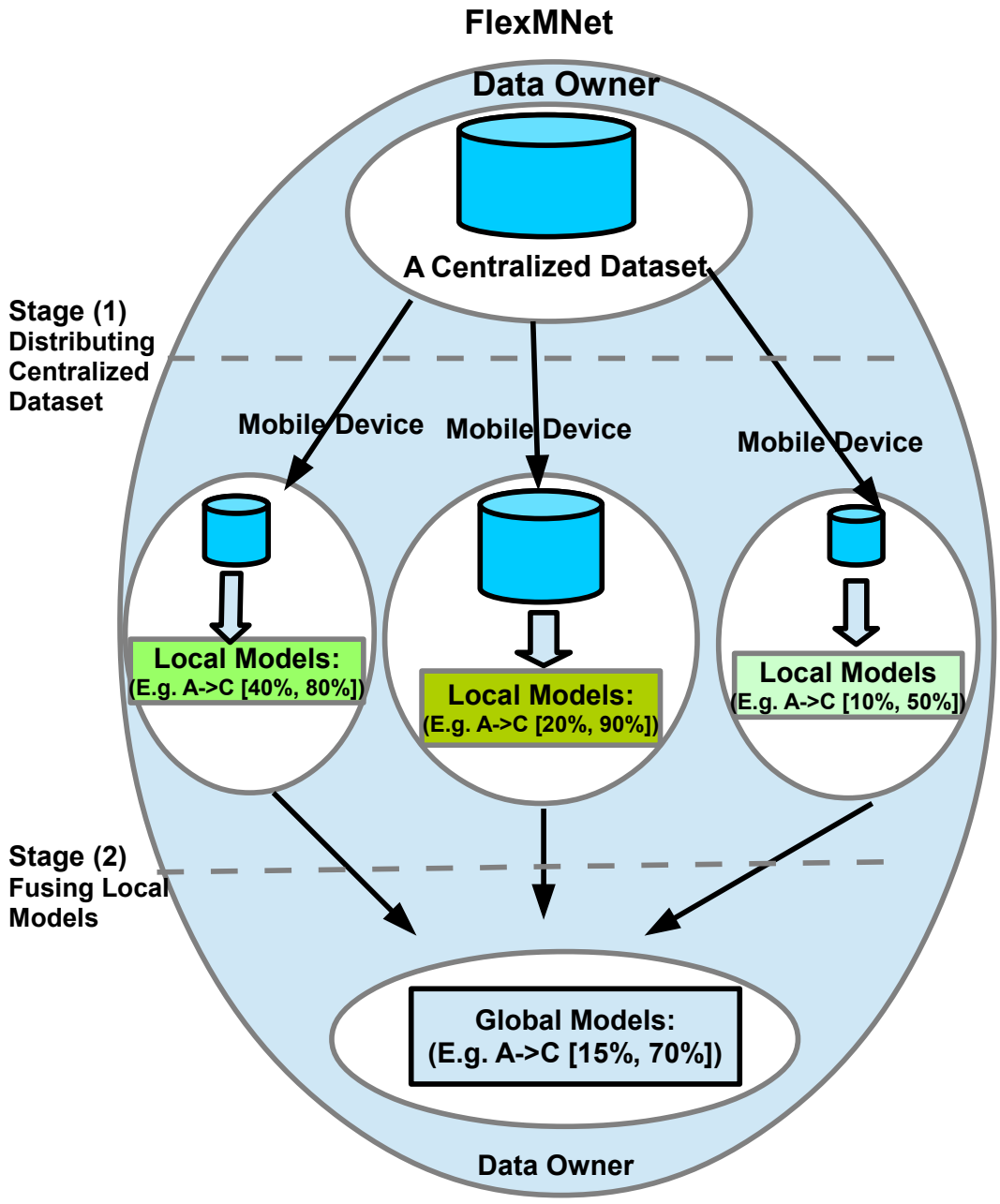

Figure 4.1. Typical procedure of association rules mining

Within the two phases, I/O cost within and among sites are regarded as the major reason to degrade the performance of a DDM algorithm (Otey et al. 2003). To reduce the I/O costs in distributed ARM, two approaches are normally applied: reducing the times of database scan and paralleling the mining task or data set (Ashrafi et al. 2004, Belbachir & Belbachir 2012).

4.2 Ranking Technique for a Distributed Data Processing Algorithm

To determine a data processing algorithm, this chapter proposes a ranking technique based on three criteria:

Criterion (1): *LightWeight* refers to the lightweight level based on the complexity of the coding of an algorithm. This criterion is calculated by the memory space that an algorithm needs to copy into MDevs for computing purposes. The number of independent functions are measured by the number of their functional lines in the coding of an algorithm. For example, a simple procedure that sums a list of numbers:

```
1: procedure sum(list)
2:     total = 0
3:     For i from 0 to length(list)-1
4:         total += list[i]
5:     EndFor
6:     return total
```

There are 6 lines in the procedure, line 5 is the jump out of the For Loop, which is not functional and independent. In this example, the value of Criterion (1) is 5.

Criterion (2): *NoOfGlobalScans* refers to the number of global data set scans. This criterion is calculated by the number of global data set scans during the entire data processing procedure.

Criterion (3): *Communication* refers to the Cost of communication. This criterion is calculated by two methods:

(1) Counting the occurrences of communications according to the steps of the processing algorithm. For example, one step in an algorithm

is *diffuse the data set to n MDevs*, the occurrences of communications between devices is counted as 1 instead of n because this is only one step and happens in parallel;

(2) Estimating the time for all *External Device Communication* .

The three criteria are given values of *Weight1*, *Weight2* and *Weight3* (valued from 1% to 100%) respectively by DO. The three weights reflect the importance of each criterion that DO specifies. A constraint applied on the three weights is $Weight1 + Weight2 + Weight3 = 100\%$. The rank of an algorithm is finally given a value by the Equation 4.1:

$$\begin{aligned} Rank\ of\ an\ algorithm = {} & LightWeight * Weight1 \\ & + NoOfGlobalScans * Weight2 + Communication * Weight1 \end{aligned} \quad (4.1)$$

Normally, large numbers are generated in calculating*LightWeight* while small numbers are in calculating the *NoOfGlobalScans*. In the situation that a DO regards *LightWeight* as a minor factor to affect the ranking result, accordingly a smaller Weight value is given to *LightWeight*. However, because *LightWeight* is calculated as a large number, it will apply a significant effect to the ranking result. In order to solve this issue, when m algorithms (A_1, A_2, ... , A_t, ... , A_m) are ranked, the values of three parameters *LightWeight*, *NoOfGlobalScans* and *Communication* are normalised for algorithm A_t by applying the Equations 4.2, 4.3 and 4.4.

$$LightWeight_N\ of\ A_t = \frac{LightWeight\ of\ A_t}{\sum_{i=1}^{m} LightWeight\ of\ A_i} \times 10 \quad (4.2)$$

$$NoOfGlobalScans_N\ of\ A_t = \frac{NoOfGlobalScans\ of\ A_t}{\sum_{i=1}^{m} NoOfGlobalScans\ of\ A_i} \times 10 \quad (4.3)$$

$$Communication_N\ of\ A_t = \frac{Communication\ of\ A_t}{\sum_{i=1}^{m} Communication\ of\ A_i} \times 10 \quad (4.4)$$

where $LightWeight_N$, $NoOfGlobalScans_N$ and $Communication_N$ indicate the values of *LightWeight*, *NoOfGlobalScans* and *Communication* respectively after normalisation.

4.3 Examples of using Ranking Technique

To demonstrate the usability of the Ranking Technique, two distributed ARM algorithms are used as examples: *ODAM* (Ashrafi et al. 2004) and *Partition* (Belbachir & Belbachir 2012). When calculating the value for Criterion (3) for the two examples algorithms, the first method, counting *Occurrences of communications*, is used.

A scenario that the two algorithms are working in is assumed with the following features;

(1) Overall computing capabilities of the MDevs involving into the FlexMNet are average;

(2) DO has small-medium volume of data;

(3) Network quality of the FlexMNet is poor, which means the latency of communications between MDevs is larger than average;

Therefore, assuming the DO gives *Weight1* as 30%, *Weight2* as 10% and *Weight3* as 60%.

External Device Communication for a distributed data processing algorithm is essential. However, every function conducting external communication is written in various codes without revealing in their publication . In order to count the number of independent functions in these two example algorithms, an assumption is made that any function conducting External Device Communication is taken account of 3 functional lines. The 3 functional lines include 3 functions; 1) connecting to a communication channel; 2) sending or receiving messages; and 3) leaving their communication channel. Taking an example of the function in ODAM algorithm, *send_to_receiver (C_2)* is

regarded as having 3 functional lines when counting the number of independent functions for the algorithm. Another assumption that needs to be made when ranking these example algorithms is that the maximum *N* for *N-frequent itemsets* is 4, that is if an algorithm needs to generate *N-frequent itemsets* by *N* iterations, it needs to execute the relevant codes by 4 times.

4.3.1 Ranking Partition Algorithm

The original version of the Partition algorithm was proposed in (Savasere et al. 1995) to find association rules in a large data set . It divides the database into horizontal partitions of the same size, where the size of each partition depends on the memory size and the average size of the transactions. It is executed on each subset of transactions (Partition) independently, and in the first scanning produces a set of frequent local itemsets for each partition. All sets of frequent local itemsets then unify as a set of global candidates. To obtain the total support for these itemsets, a second scanning of the database is necessary. Although the algorithm is called *Partition*, its entire computation is completed within one machine. However, as the algorithm was designed to process a partitioned data set, it is easy to be extended to its parallel/distributed version. One parallel version of the Partition algorithm is proposed in (Belbachir & Belbachir 2012).

This parallel approach adopted a set of N sites (clients) managed by a site called Coordinator (server). The process of their approach has four phases.

- *Phase 1*: each processor has a partition. It applies the procedure gen_large_itemsets (see Figure 4.2) on local data in order to identify the set of frequent itemsets locally L_i, then it sends the result to the coordinator. As the size of the local data in all nodes is approximately equal, this phase will take approximately equal time in all nodes realising a balancing of charge.
- *Phase 2*: after receiving the results of phase 1, the coordinator will undertake to identify the set of global candidates C^G, making the union of all sets of frequent itemsets locally in each of the processor (treatment realised in merged phase of

the Partition algorithm). The result is diffused to all processors. At the end of this phase, all nodes will have exactly the same set of candidate itemsets.

- *Phase 3*: each processor determines the support for all itemsets in C^G, then sends them to the Coordinator.
- *Phase 4*: after having received the result of phase 3, the Coordinator will undertake to identify the set of global frequent itemsets making the summation of the supports for each itemset.

Line No.	Pseudocode of Algorithm
	procedure gen_large_itemsets(p: database partition)
1)	L_1^P = {large 1-itemsets along with their tidlists}
2)	**for** ($k = 2$; $L_k^P \neq \emptyset$; k++) **do begin**
3)	**forall** itemsets $l_1 \in L_{k-1}^P$ **do begin**
4)	**forall** itemsets $l_2 \in L_{k-1}^P$ **do begin**
5)	**if** $l_1[1] = l_2[1] \wedge l_1[2] = l_2[2] \wedge \cdots \wedge l_1[k-2] = l_2[k-2] \wedge l_1[k-1] < l_2[k-1]$ **then**
6)	$c = l_1[1] \cdot l_1[2] \cdots l_1[k-1] \cdot l_2[k-1]$
7)	**if** c cannot be pruned **then**
8)	$c.\text{tidlist} = l_1.\text{tidlist} \cap l_2.\text{tidlist}$
9)	**if** $\lvert c.\text{tidlist} \rvert / \lvert p \rvert \geq minSup$ **then**
10)	$L_k^p = L_k^p \cup \{c\}$
11)	**end**
12)	**end**
13)	**end**
14)	**return** $U_k^1(L_k^p)$

Figure 4.2. Procedure gen_large_itemsets (Belbachir & Belbachir 2012)

The number of independent functions is measured by the number of their functional lines in the codes of an algorithm, which has been mentioned before. Phase 1 and 3 in Figure 4.3 happen in each MDev and Phase 2 and 4 happen in the Coordinator. Accordingly, independent functions for calculating the value of *LightWeight* for parallel Partition are represented in Phase 1 and 3;

Phase 1: Line 7 (3 lines because it represent a communication function) in Figure 4.3;

Line No.	Pseudocode of Algorithm
1)	**Entry**: Database D, the number of sites P, the minimal support minsup
2)	**Exit**: Set of frequent itemsets L^G
3)	Coordinator: partitions the data base horizontally to N partitions and assigns them to the different sites.
4)	**Phase 1** :// in parallel each site generates the frequent local itemsets
5)	**For each** site I=1…….P **do in parallel**
6)	$L^{(i)}$ = **gen-Iarge-itemsets** (P_i) // Find the set of frequent local itemsets;
7)	// Each site send $L^{(i)}$ to coordinator to calculate the set of global candidates ;
8)	**End of parallel**
9)	**Phase 2** : // the coordinator calculate the set of global candidate itemsets
10)	Coordinator: $C^G = U_{i=1...p} L^{(i)}$;
11)	// Diffuse C^G to all processors;
12)	**Phase 3** : // count the supports of global candidates
13)	**For each** site I= 1.....P **do in parallel**
14)	**For each** $c \in C^G$ **do**
15)	$c.compter^{(i)}$ = gen-count (c , p_i) ;
16)	// Send the results to coordinator;
17)	**End**
18)	**End of parallel**
19)	**Phase 4** : // the coordinator calculate the global supports of global candidates
20)	Coordinator**: For each** $c \in C^G$ **do**
21)	$c.compter = \sum_{i=1}^{p} c.compter^{(i)}$ // combine accounts;
22)	**Return** $L^G = c \in C^G \mid \sum_{i=1}^{p} c.compter^{(i)} \geq min\_sup$;
23)	**End**

Figure 4.3. The parallel algorithm of Partition (Belbachir & Belbachir 2012)

Line 1-5 (5 lines) and Line 7, 9 and 14 (3 lines) in Figure 4.2, which are in the gen-large-itemset (p_i) function (Line 6 in Figure 4.3).

Phase 3: Line 14 and 15 (2 lines), Line 16 (3 lines because it represent a communication function) and after Line 12 (3 lines because one communication function to receive the global candidates from the Coordinator although the communication function has not been written in the algorithm).

Consequently, the number of independent functions is $3+8=11$ in Phase 1 and $2+3+3=8$ in Phase 3. In total, the value of *LightWeight* is 19.

3 global dataset scans occur when 1) sending the partitions of the original dataset to each site; 2) calculating the set of global candidate itemsets in Phase 2; and 3) calculating the global support s of global candidates in Phase 4. In total, the value of *NoOfGlobalScans* is 3.

External Device Communication happens in Line 3, Line 7, Line 12 and Line 16 (4 times in total). Accordingly, the value of the Occurrences of *Communication* for parallel Partition is 4.

4.3.2 Ranking ODAM Algorithm

The pseudocode of the Optimized distributed ARM algorithm (ODAM) (Ashrafi et al. 2004) is illustrated in Figure 4.4 and the line numbers of independent functions of it are added in the figure.

> ODAM first computes support counts of 1-itemsets from each site. It then broadcasts those itemsets to other sites and discovers the global frequent 1-itemsets. Subsequently, each site generates candidate 2-itemsets and computes their support counts. At the same time, ODAM also eliminates all globally infrequent 1-itemsets from every transaction and inserts the new transaction into memory. While inserting the new transaction, it checks whether that transaction is already in the memory. If it is, ODAM increases that transaction's counter by one. Otherwise, it inserts the transaction with

> a count equal to one into the main memory. After generating support counts of candidate 2-itemsets at each site, ODAM generates the globally frequent 2-itemsets. It then iterates through the main memory and generates the support counts of candidate itemsets of respective length. Next, it generates the globally frequent itemsets of that respective length by broadcasting the support counts of candidate itemsets after every pass (Ashrafi et al. 2004).

Independent functions for ODAM are represented in Line 1-6 (6 lines) , Line 8-15 (7 lines) and Line 17 (1 line). Additionally, there are 3 communication functions in the algorithm, two *send_to_receiver (C_2)* s (Line 7 and Line 16) and one *receive_from_receiver (F_g)* (Line 8), each of which is regarded as having 3 functional lines. Consequently, the number of independent functions is $6+7+1+3*3=23$, the value of *LightWeight* thus is 23.

1 entire dataset scan occurs when the partitions of the original dataset are sent to each site, the value of *NoOfGlobalScans* thus is 1.

External Device Communication happens in Line 7, Line 8 and Line 16 (3 times in total). In terms of the assumption made before that the algorithm needs to execute 4 times in order to generate *4-frequent itemsets*, the value of the Occurrences of *Communication* for ODAM is $3*4=12$.

4.3.3 Results

For *Partition* algorithm, the values of the ranking factors multiply their *Weight*s are illustrated in Table 4.1 and the total ranks are calculated in Equation 4.5 by applying Equation 4.1.

The total ranks are calculated by applying Equation 4.1, which is

$$1.425+0.75+1.5=3.675 \tag{4.5}$$

For *ODAM* algorithm, the values of the ranking factors multiply their *Weight*s are illustrated in Table 4.2 and the total ranks are calculated in Equation 4.6 by applying Equation 4.1.

Line No.	Pseudocode of Algorithm
1)	NF = {Non-frequent global 1-itemset}
2)	for all transaction $t \in D$ {
3)	for all 2-subsets *s* of *t*
4)	if ($s \in C2$) *s*.++;
5)	t ‘ = delete_ nonfrequent_items (t);
6)	Table .add (t’)
	}
7)	send_ to_ receiver (C_2);
	/*Global Frequent support counts from receiver */
8)	F_2 = receive_from_receiver (F_g) ;
9)	C_3 = {Candidate itemset};
10)	T = Table.getTrasactions ( ) ; k = 3;
11)	while (Ck ≠ {}) {
12)	for all transaction t ∈ T
13)	for all k-subsets *s* of *t*
14)	if (s ∈ *C*) *s*.sup ++;
15)	k ++;
16)	Send_to _receiver (C_k) ;
17)	/*Generating candicate Itemset of *k + 1* pass*/
18)	C_k +1 = {Candidate itemset};
	}

Figure 4.4. The pseudocode of ODAM algorithm (Ashrafi et al. 2004)

Table 4.1. Ranks for Partition algorithm

Item	Ranks=Value (1-10) * Weight(1%-100%)
$LightWeight_N$ * *Weight1*	$\frac{19}{19+21} \times 10 \times 30\% = 1.425$
$NoOfGlobalScans_N$ * *Weight2*	$\frac{3}{3+1} \times 10 \times 10\% = 0.75$
$Communication_{[1]}$ * *Weight3*	$\frac{4}{4+12} \times 10 \times 60\% = 1.5$
[1]: *Communication* in this table indicates the value of the Occurrences of Communication after being normalised	

Table 4.2. Ranks for ODAM algorithm

Item	Ranks=Value (1-10) * Weight (1%-100%)
$LightWeight_N$ * *Weight1*	$\frac{21}{19+21} \times 10 \times 30\% = 1.575$
$NoOfGlobalScans_N$ * *Weight2*	$\frac{1}{3+1} \times 10 \times 10\% = 0.25$
$Communication_{[1]}$ * *Weight3*	$\frac{12}{4+12} \times 10 \times 60\% = 4.5$
[1]: *Communication* in this table indicates the value of the Occurrences of Communication after being normalised	

$$1.575 + 0.25 + 4.5 = 6.325 \tag{4.6}$$

When the DO specifies the values of *Weight1*, *Weight2* and *Weight3* as 30%, 10% and 60%, the *Partition* algorithm (ranked as 3.675) is a better algorithm to process the data than the *ODAM* algorithm (ranked as 6.325) for a scenario in which

(1) overall computing capabilities of the MDevs involving into the FlexMNet are average;

(2) the DO has small-medium volume of data set; and

(3) network quality of the FlexMNet is poor, which means the latency of communications between MDevs is larger than average.

4.4 Summary

In FlexMNet, algorithms that process data sets are regarded as one of the factors that affect the performance of a processing task. Based on this understanding, this chapter discusses how the algorithms affect performance from the perspectives of communication between devices, and the limited resources of MDevs. Furthermore, a number of requirements that the algorithms need to meet in order to reduce the impact on performance have been proposed, including;

(1) Light-weight algorithms in both coding and complexity in order to minimise the possession of computing resources in MDevs;

(2) Algorithms designed with less External Device Communication but more Internal Device Communication; and

(3) Minimising the number of entire data set scans.

Finally, this chapter presents a Ranking technique in terms of the requirements in order to assist the medium data processing tasks over FlexMNet to select the optimised algorithm for the data sets. Two example algorithms, *Partition* and *ODAM*, demonstrate the usage of the Ranking technique in a hypothetical scenario.

Chapter 5

Experiment of Performance Improvement

5.1 Aims and Setup of the Experiment

This chapter describes the experiment in order to demonstrate the application and study the accuracy of the prediction strategy. The results from the experiment will be used to improve the performance for a medium data processing task over FlexMNet.

The experiment is conducted in a simulated environment where six MDevs are connected by Wi-Fi signals within a sphere that the radius of it is about 15 metres. The six MDevs have various computing capabilities (see in Table 5.1). The job of the experiment is to predict a group of MDevs with the quickest Processing Time in order to process a given data set. The experimental environment, the data set and the Med-data algorithm are described respectively in the next sections.

5.1.1 Experimental Data Set

A DDM algorithm is selected by the experiment as an example algorithm for processing a medium data set over FlexMNet. Details of the algorithm is given in a later section. A relational data set is processed by this algorithm as an example. A relational data set is a collection of data items organized as a set of formally described tables (Codd 1970). The experimental data set uses 8000 records of a relational synthetic data set, which contains 3 attributes: record ID, location (coordinate x, coordinate y) and a short paragraph of description. The size of the data set is roughly 200 MBs.

The experimental data set managed by it DO is stored in a Dell laptop. The experimental data set will be horizontally partitioned and the partitions will be sent to the MDevs according to the computing capability of each MDev. Therefore, the parallel clustering algorithm selected by the experiment is required to be able to process homogeneous distributed data set. The data set has been cleaned to meet the requirements for conducting the experiment.

5.1.2 Experimental Environment

The six MDevs which are equipped with Android operating systems are available for this experiment. A sample data set which is 5 MBs and copied from the experimental data set is used to obtain the computing capability of each MDev before processing the experimental data set. Table 5.1 lists the features of the MDevs: the specifications, the Processing Time, the Computation Time and the number of CUs.

The Computation Time is measured by different approaches for the experimental data set and the sample data set;

- for the sample data set: each MDev processes the sample data set by squaring the value of coordinate x in each record;
- for the experimental data set, each MDev processes the experimental data set by implementing the Med-data algorithm that has been copied into each MDev.

The Processing Time includes three parts:

Part (1) Time taken to send the sample or experimental data from the DO to each available MDev;

Part (2) The Computation Time;

Part (3) Time taken to send the result back to the DO.

In the Table, the method used to convert the Processing Time to the number of CUs is shown in the Equation 5.1.

$$\frac{\textit{The minimum Processing Time in the six MDevs}}{\textit{The Processing Time of a MDev}} * 100 \tag{5.1}$$

For example, the Processing Time of the device A is 4.88 mins and the minimum Processing Time is 3.09 mins, which is given by the device B. According to Equation 5.1, the number of CUs for the device A is calculated as $\frac{3.09}{4.88} * 100 = 63$. Apparently, the computing capability of the device B is 100 CUs.

Table 5.1. Specifications of participating MDevs

MDev[1]	Processor	RAM	Available Space	Battery Duration[2]	Connection	Processing Time	Computation Time[3]	Number of CUs
A. Samsung Tablet	Quad-core 1.2 GHz	2 GBs	6.4 GBs	24 hrs (8 hrs)	Wi-Fi (802.11 a/b/g/n/ac) 2.4G + 5GHz	4.88 mins	1.32 mins	63 CUs
B. OPPO R9S Smart-Phone	Octa-core 2.0 GHz	4 GBs	50 GBs	5 hrs (1.6 hr)	Wi-Fi 802.11 a/b/g/n/ac, dual-band, Wi-Fi Direct	3.09 mins	1.39 mins	100 CUs
C. SamSung Smart-Phone Galaxy 2	1.2 GHz Dual Core	1 GB	1.21GB	3.4 hrs (1hr)	USB 2.0/802.11 a/g/b/gn (2.4GHz /5GHz), Wi-Fi Direct	24.22 mins	7.83 mins	13 CUs
D. AUSA Tablet	Quad-core 1.2 GHz	1 GB	4.6 GBs	20 hrs (6hr)	Wi-Fi 802.11 b/g/n	25.2 mins	16.22 mins	12 CUs
E. SamSung Smart-Phone Galaxy 4	Quad-core 2.3 GHz	2 GBs	3.96 GBs	6.8 hrs (2hr)	Wi - Fi 802.11 a/b/g/n/ac, 4G	6.37 mins	4.59 mins	49 CUs
F. HTC SmartPhone	Quad-core, 1.7 GHz	2 GBs	1.4 GB	4 hrs (1.2 hr)	802.11 a, b, g, n, dual-band	10.92 mins	5.63 mins	28 CUs

[1] : Assuming that these participating MDevs are available during the entire process

[2] : When various MDevs are running the same task, their battery powers are consumed differently depending on the performance of their battery. Moreover, when a MDev is processing a computing-intensive task, its battery power is consumed quicker than when it is idle. Finally, none of the device owners would like their battery gets flat after they complete the task. For these three reasons, this experiment considers 1/3 of their battery life, which is shown in a round bracket.

[3]: Computation Time does not include the Transmission Time .

5.2 The Algorithm

This experiment chooses a parallel sufficient statistics approach for the distributed K-Means clustering technique, which was proposed in (Forman & Zhang 2000), as the Med-data algorithm for processing the dataset. The reason for choosing this algorithm is that it meets the requirements of this experiment, such as the small communications between MDevs and the insufficient storage space in MDevs.

5.2.1 K-Means Algorithm and Its Parallel Version

The K-means algorithm is a prototype-based partitional clustering technique that aims to find a user-specified number of clusters (*K*), which are represented by their centroids, the centres of the clusters (Tan, Steinbach & Kumar 2006). The K-means algorithm needs the number of clusters *k* to be decided before conducting the steps which are summarised in Table 5.2:

Table 5.2. Steps of K-means

Step No.	Tasks
(1)	Set *K* points as initial the centroids;
(2)	Assign each data point to its closest cluster based on the calculation of the square of the Euclidean distance between each point and its centroid;
(3)	Recompute the centroid of each cluster by calculating the mean of all data points belonging to that cluster;
(4)	Repeat steps 2-3 until centroids do not change.

A parallel version of K-Means was proposed in (Forman & Zhang 2000) and the algorithm of this version is adopted to implement the experiment in this chapter. Two variations are made on this approach in order to adapt the environment of the experiment. Firstly, computers used in their approach have been changed to MDevs. Secondly, as the environment of this experiment is a group of MDevs, an algorithm to independently process the distributed data set needs to be copied in each single MDev. For this reason, one more task, which is *a distributed K-means algorithm to produce intermediate results is copied to each MDev*, has been added into *Step 1*.

Step 1 : Arbitrarily distribute the N elements of the data set S to the local memories of a set of P MDevs; a distributed K-means algorithm to produce intermediate results is copied to each MDev.

Step 2 : Pick the estimated K initial centre locations M by any schema, such as a random sample. A coherent copy is kept on each MDev throughout the computation.

Step 3 : Iterate:

3. (a) Each MDev independently computes its contribution to a set of globally Sufficient Statistics, which includes information for computing the performance function;

3. (b) The sufficient statistics are globally reduced (summation across processors), and then the results are broadcast back back to all MDevs, and

3. (c) Independent local computation adjusts the centre location estimates. The results are identical on each MDev and are exactly the same as the uniprocessor sequential algorithm would produce.

Step 4 : Stop when the performance function converges, or after a fixed number of iterations.

Step 5 : Output the K centres M.

After the data set has been transferred into each MDev, communications which occur between MDevs in each step of the algorithm, as shown in Table 5.3.

Table 5.3. Tasks with significant time cost in each step

Step No.	Communications between MDevs
1	Transferring data partitions to each MDev, which results in the major communication cost over Network.
2	No communication.
3.a	Calculating the Euclidean Distance of each elements;
3.b	Computing global reduction; Broadcasting the intermediate results.
3.c	Calculating the distance of each elements to adjust the centre.
4	Transmitting all the centres to the DO in order to integrate the local results.

In order to demonstrate the usage of the prediction strategy in a clearer way, the above algorithm is simplified by three steps:

Step (i) Generate local clusters by the centralized K-means algorithm in each MDev; If given any dataset, iterations in Step 3 in the above algorithm cause a large amount of communication to occur between devices which does not benefit result comparison. Therefore, this experiment chooses to set a fixed number of iterations, which is 2.

Step (ii) Merge the local clusters into global ones by using Step 3; if the task of merging local clusters happens in the same MDev as the original data, the DO receives the final result; otherwise, progress to Step (iii);

Step (iii) The global clusters as a final result are sent to the DO.

Before the processing starts, the values of the following parameters need to be provided as the input of the Med-data algorithm:

- Time duration that the MDevs contributes its computational resources to the DO;
- Throughput of the network and time duration when the network remains throughput-wise steady in the previous hour before the processing starts;
- Computing capability of each participating MDev;

The algorithm is implemented in Java and in two versions: (1) the laptop version on Eclipse platform; and (2) the smartphone and tablet version on Android Studio platform.

5.3 Procedure and Results

After determining the data set and the Med-data algorithm, the next step is to select the candidate groups from the six available MDevs.

According to the mathematical combination, six MDevs can form $\binom{6}{1} + \binom{6}{2} + \binom{6}{3} + \binom{6}{4} + \binom{6}{6} + \binom{6}{6} = 6 + 15 + 20 + 15 + 6 + 1 = 63$ MDevs groups. Apparently, it is impossible to implement the experiment over all groups due to the large number of them. In order to reduce the number, the majority of the groups are pruned by applying several rules discussed in the prediction strategy described in Chapter 3. Firstly, the groups with one MDev and the groups with all six MDevs, are pruned. Secondly, any groups with MDevs C and D, are pruned because both C and D have fairly small number of CUs. Thirdly, the device B has a low battery duration (1.6 hr) although it has a powerful computing capability based on its high number of CUs. Additionally, the device F does not have a powerful computing capability and its battery duration is low. Accordingly, the DO should not send the devices B and F the data partitions, the sizes of which are beyond what they can process within their battery duration. Finally, it is noticed that the device E has a short Transmission Time (Processing Time - Computation Time) , which is 6.37 - 4.59 = 1.78 mins, compared to the other devices. However, it does not process the sample data set very fast (4.59 mins). This phenomenon indicates that the device E performs better in externally transmitting the data than in internally processing the data.

After pruning the groups with one MDev and all six MDevs and the groups with the devices C and D, the rest of the groups are listed in Table 5.4. In the table, the Group 5 needs to be further pruned because of the battery limitations of both devices B and E. The Groups 4, 6 and 10 need to be pruned because the device E cannot process a larger partition of the data set for its groups due to its slow computing speed. The rest

of the 6 groups, Groups 1, 2, 3, 7, 8 and 9, are the candidate groups.

Table 5.4. Candidate groups

Group	MDevs in Each Candidate Group
Group 1	A, B
Group 2	A, E
Group 3	A, F
Group 4	B, E
Group 5	B, F
Group 6	E, F
Group 7	A, B, E
Group 8	A, B, F
Group 9	A, E, F
Group 10	B, E, F

The next step is predicting which group in the five groups can provide the best Processing Time according to the prediction strategy proposed in Algorithm 1 in Chapter 3. As this experiment does not consider the memory usage in the MDevs, only Processing Time (in this experiment, Processing Time is replaced by the number of CUs) of each group is calculated. After sorting out the Processing Time of all candidate groups, Group 7 is predicted as the group with the best performance due to its highest number of CUs (63+100+49 =212) compared to the others groups (163, 112, 91, 191 and 140 respectively).

In order to verify the prediction result, the experimental data set is processed by all candidate groups using the Med-data algorithm. The Processing Time of each group is recorded. Additionally, the data set and temporary results stored in the cache and memory space of the MDevs need to be cleaned after the execution of each candidate group. The aim of doing so is to ensure that the cache and memory space of the MDevs matches their specifications provided in Table 5.1 when each candidate group starts its processing.

The Processing Time produced by each candidate group is listed in Table 5.5.

Table 5.5. Processing Time of each candidate group

Group No.	Processing Time (mins)
Group 1	63 mins
Group 2	59 mins
Group 3	78 mins
Group 7	57 mins
Group 8	82 mins
Group 9	54 mins

5.4 Discussions and Summary

The result in Table 5.5 shows that Group 9 provides the shortest Processing Time compared to the other groups. Group 7 also produces a compelling and acceptable Processing Time although it does not produce the shortest one as predicted. The reasons for the unmatched results are various. They can be the unpredictable and changeable wireless network latency, the Apps or programmes running in the background of the MDevs and, other factors which will affect the performance but haven't been considered by the prediction strategy. However, there is always a conflict between the complexity of designing a strategy and the efficiency of implementing the strategy. When more factors are considered in a prediction strategy, the increased complexity of the strategy makes MDevs difficult to apply it due to the limited computing resources in MDevs.

In summary, the experiment demonstrates the usage of the prediction strategy which aims to help a DO make a quick decision on selecting a number of MDevs with the best performance. The results of the experiment summarise and point out the future work for improving the prediction strategy proposed in Chapter 3.

Chapter 6

Medium Data Processing Framework over FlexMNet

6.1 Introduction

Chapter 3 describes the proposed prediction strategy to improve the performance of a distributed data processing algorithm over a FlexMNet. The strategy comprises three major steps: determining Med-data algorithms; participating MDevs selection; and hiding sensitive information, if DOs have privacy concerns. An over-arching framework is proposed comprising these three steps in order to process medium data over a FlexMNet. The proposal of a framework also benefits the application of a general data processing algorithm over FlexMNet. To illustrate and discuss this framework, distributed DM strategies are adopted as an example for processing medium data.

Distributed medium and large data processing frameworks or systems have been initially studied based on the needs of large volumes of data (Park & Kargupta 2003, Wang et al. 2003, Tsoumakas & Vlahavas 2008). Those needs are mainly from financial organisations where the growth of data exceeds their capability for processing. These organisations often have powerful computers and advanced data processing strategies. However, these hardware and software conditions are not applicable to many industrial or commercial scenarios which require DDM. One example scenario is local MDevs which have no access to supercomputers or computing clouds via the Internet to apply computing-intensive processes to their data. In addition, there are remote areas which have no commercial communications network coverage which can utilise mesh networks such as those being developed by the Serval project (Gardner-Stephen & Challans 2010). Therefore, conducting computing-intensive data processing in this sort of network, using any available physical device, has significant benefit for both commercial and humanitarian organisations.

6.2 Medium Data Processing Framework for a FlexMNet

The medium data processing framework is designed by considering the major features of a FlexMNet;

- the topology of the network is dynamic and self-configuring;
- MDevs voluntarily contribute their computational resources to a data processing task;
- MDevs have various computational resources, such as algorithms to process a set of data, memory space, and privacy protection algorithms to hide sensitive information.
- MDevs register their available computational resources in the network and they can be searched as available devices when a DO has a processing request; and
- the computational capabilities of the various MDevs are not identical.

The framework over FlexMNet needs to have several basic functionalities, such as *computing*, *generating results* and *collaboration between MDevs*. These functionalities are achieved by fusing the existing research from the following fields:

- *Distributed and parallel computing*, which contributes to achieving the computing capability in FlexMNet;
- *Agent-based and mobile based DM*, in which intelligence exchange benefits the collaboration work and *generating results*;
- *Social group work*, some features of which contribute to design *collaboration between MDevs*. A social group work is a system through which individuals in groups in a social agency setting are helped by a worker who guides their interaction through group activities. Consequently, the individuals relate to others and share their experience in accordance with their needs and capacities (Konopka 1963).

In terms of the major features of FlexMNet and contributions from the existing research fields, the characteristics of the framework are defined as;

- it comprises contributors;
- it works in a distributed environment, in which a processing task is implemented with other unknown cooperative objects if needed;
- the contributors are collaborative;
- the contributors have the intelligence to complete part of a specific task;
- the contributors within it have dynamic instantiation; and
- the contributors within it are function-based (for example, algorithm contributor, computing contributor and resource contributor).

In the framework, computing tasks are conducted by CCs, see Chapter 3, which is a MDev that is available to participate in and complete a data processing task launched by a DO. The aim of the framework is to organise a number of Contributors to work collaboratively and efficiently.

6.2.1 Cooperative Working Scenario

A *Cooperative Working Scenario* focusing on a small organisation of MDevs is described as follows:

- The group of devices are within a sphere covered by a local wireless network, e.g. a Wi-Fi network.
- MDevs voluntarily register as member devices to contribute to and benefit from the services that a trusted collaborative network provides.
- There is trust within the group. Each group member trusts all other members to view and store their data.
- Each group member can use at least one device to upload their data.
- All group members are collaborative. Each device has the capacity to provide computing power and/or intelligence for decision making.
- After a device commits to provide its computing capability to a data processing task, the available time of the device to the process is set (e.g. one hour)

and the task is given the highest priority. This guarantees the availability of resources needed. If another process in the device requires resources, such as memory or the CPU, currently occupied by the processing task, this process will be blocked/suspended until the collaborative task releases the resources.

- While processing a data set, transmission of data between devices is kept to a minimum.

The framework is designed with the following assumptions:

- Time taken for transferring messages between MDevs should be acceptable in terms of DO's requirements. For example, a museum visitor needs a number of images he/she owns to be processed within one hour while he/she is still in the museum.
- Individual privacy is not a primary concern for each participating MDev owner because the collaborative data processing task is undertaken within a trusted or semi-trusted membership network.

6.2.2 Framework Components

All Contributors are transient not persistent, which means they only exist while they are providing a service for a data processing task. *Contributor*s used in the framework include the following types;

- A server controls membership of the flexible network and maintains the registration information of each of the Contributors. The computing capacity required for the server depends on the number of contributors to the network. A smartphone or personal computer is able to maintain this information for several hundred devices.
- *Task Contributor (TC)* owns one data processing task, which can

  i) broadcast a Processing Request (PR); and

  ii) gather information sent from a Resource Contributor(s) as a Request Message and send the Message to Algorithm Facilitator Contributor.

A TC must be applied for all data processing requirements, whether it is for a DO or not.

- *Computing Contributor* implements computation tasks, which has been defined in Chapter 3. A CC can:

  i) cooperate with other CCs for a data processing task; and

  ii) copy algorithm scripts sent by an Algorithm Contributor (AC) to its own device when processing a task and deleting the scripts after the task completes.

- *Computing Facilitator Contributor (CFC)* organises and manipulates a dynamic Computing Contributor Tree and deploys the computing load to a number of CCs. A MDev can be both CFC and CC.

- *Algorithm Contributor (AC)* owns one data processing algorithm that refers to a specific domain and data type. The data processing algorithm can be either distributed or centralised. An Algorithm *Contributor* can:

  i) apply to be an Algorithm Facilitator Contributor if it has sufficient computing resources to undertake the duties of an Algorithm Facilitator Contributor; and

  ii) share the algorithms stored in itself when it registered as an AC.

- *Algorithm Facilitator Contributor (AFC)* assists the Task Contributor to find the most suitable algorithm contributed by available ACs. An AFC can:

  (a) receive and process Request Messages from a TC;

  (b) broadcast the data processing request to the surrounding area in order to search ACs and organise responses from the available ACs;

  (c) communicate with the Server where an Algorithm Contributor Tree is stored in order to authenticate the membership and other information of the available ACs;

  (d) search and retrieve appropriate ACs in an Algorithm Contributor Tree according to a Processing Request;

(e) determine the most suitable algorithm codes according to 1) the requirements of the Processing Request; and 2) information sent from the Server;

(f) send the selected algorithm to a Recommendation Model to predict the group of CCs with the best performance; and

(g) receive the recommended group from the Recommendation Model and organise the relevant CCs to execute the processing task.

AFC and AC can be applied by one MDev. In the framework, an AFC undertakes the majority of workload on communication, recommendation and determination, thus its role needs to be taken by a more powerful MDev compared to other ones in a FlexMNet.

- *Resource Contributor (RC)* owns data sets or images for processing. A RC can:

  i) share its resource with other Contributors in order to assist them to complete their tasks; and

  ii) seek assistance from other Contributors to complete its own requirement.

  Therefore, RC is not applicable to some data processing requirements over FlexMNet. For example, when a DO requires a number of MDevs to process its data set in a distributed but still collaborative manner, there is no need for a RC to contributor its recourse.

States of Contributors include *Waiting*, *Ready* and *Occupied*. The State is set to *Waiting* when a Contributor sends a bid to be a Facilitator Contributor. The State is set to *Ready* when a Contributor is initialised or fails to bid as a Facilitator Contributor. The State is set to *Occupied* when a Contributor is implementing processing tasks.

6.2.3 Properties of Contributors

The following terms define the properties of the various Contributors.

- *Request Message (RM)* comprises one *Processing Request* (PR) and the location(s), size(s) and data type(s) of $0 \ldots n$ RC(s). RCs are organised in order of the size of their available computing resources. A RM is sent by the TC and received by AFC.

- *Contributor Token (CT)* comprises the type of contributor (which contains the path of the Contributor's parent nodes), *Time Stamp* and

  (i) available storage space and CPU speed and location if it is a CC;

  (ii) domains the algorithm can apply, input constraints for the algorithm and parameters needed to be set if it is an AC; and

  (iii) resource size and location and data structure if it is a RC.

- *Algorithm Contributor Tree (ACT)* stores the fundamental information of registered ACs classified by data processing tasks. Consider the example of the information stored in an AC for a DM process. DM algorithms are categorised into five classes of core mining tasks (Tan et al. 2006): Cluster Analysis, Predictive Modelling, Association Analysis and Anomaly Detection. Each class has its associated mining algorithms. The Token of an AC is located at the tree leaf of the corresponding algorithm class, see Figure 6.1. The Tokens can be added in, when available, or updated, when the information in the Contributor Token has changed, but the Contributor ID remains the same. The Token can be removed from the ACT when the ID number of a CC has changed, for example, when the AC changed its mining algorithm from DBSCAN to K-means. It is an essential for any data processing procedure.

- *Computing Contributor Tree (CCT)* stores information of registered CCs. The CCT is organized in terms of different scales of computing space. Contributor Tokens associate their Contributors with matching scales, for example, 1 Gigabyte, 10 Gigabytes or 20 Gigabytes. A Contributor Token can be added in when it is available, and removed from the CCT when it is occupied for a computing task. It is essential for any data processing procedure.

- *Resource Contributor Tree (RCT)* stores information of registered RCs. The CCT is organised in terms of different types of data resources, for example, structured data set, unstructured data set, images and videos. It is an optional component for a data processing procedure and created when RC is available.

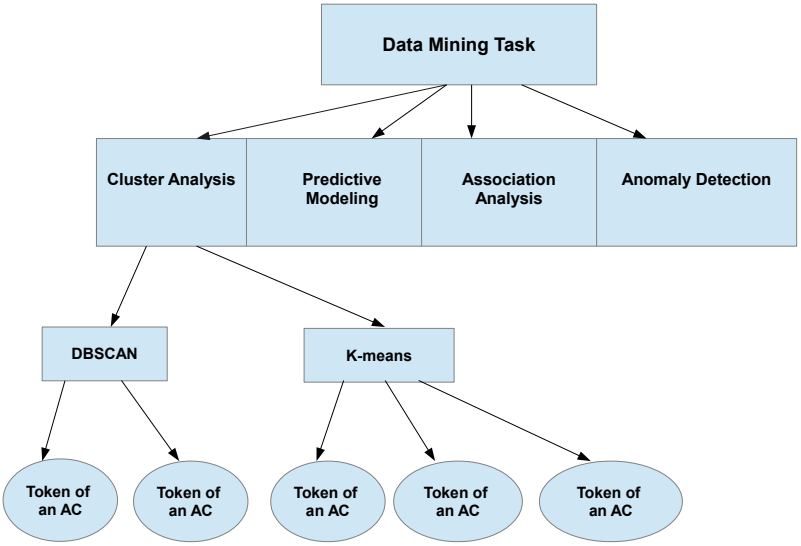

Figure 6.1. Example of Algorithm Contributor Tree

There are two approaches to creating the CCT:

(i) Create a static Tree and update it at certain time intervals. This approach is the default in the framework;

(ii) Dynamically create the Tree when the AC requires it.

The names of all components and their initials are listed in Table 6.1.

The interactions, as described, of these Components in an example of the framework over FlexMNet is illustrated in Figure 6.2.

6.3 Framework Layers

FlexMNet aims to improve the internal task process and feasibility of a small group, and has to follow certain preliminaries in order to enable control and facilitate knowledge diffusion. Communications in FlexMNet are organised into four layers. A layer serves the layer above it and is served by the layer below it.

- *Application Layer* conducts two major functionalities: 1) quickly respond to external queries, including analysing the requirements of a data processing task

Table 6.1. Framework components

Component Names	Acronyms
Algorithm Contributor	AC
Algorithm Contributor Activator	ACA
Algorithm Contributor Generator	ACG
Algorithm Contributor Request Queue	ACRQ
Algorithm Contributor Tree	ACT
Algorithm Contributor Tree Organiser	ACTO
Algorithm Facilitator Contributor	AFC
Computing Contributor	CC
Computing Contributor Activator	CCA
Computing Contributor Generator	CCG
Computing Contributor Recommender	CCR
Computing Contributor Tree	CCT
Computing Contributor Tree Organiser	CCTO
Computing Facilitator Contributor	CFC
Contributor Token	CT
Processing Request	PR
Request Message	RM
Resource Contributor	RC
Resource Contributor Activator	RCA
Resource Contributor Generator	RCG
Resource Contributor Tree Organiser	RCTO
Task Contributor	TC

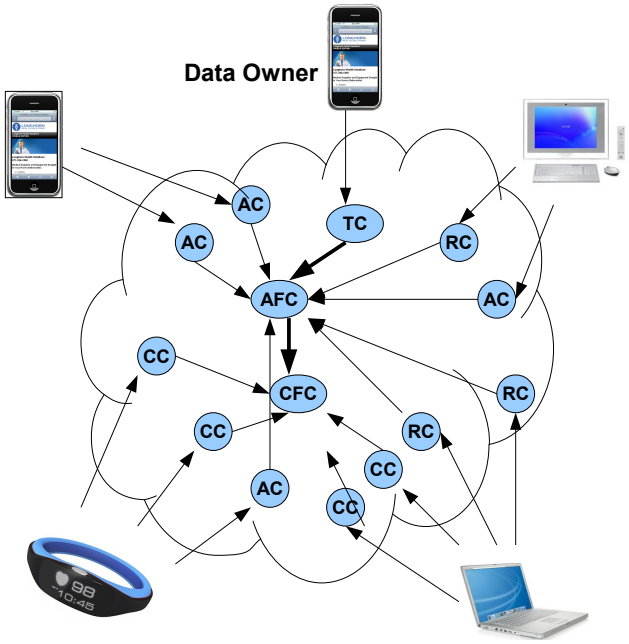

Figure 6.2. An example of the framework over FlexMNet

from user(s) and selecting a Med-data algorithm from a number of available ones; and 2) decide the format of the results then collaboratively display the result. This layer contains two major components: Task Analysing/Processing and Results Display. Existing customised task analysing/processing tools or approaches can be adopted in Task Analysing/Processing components. Existing or customised result display tools or software can be adopted in the Result Display component.

- *Recommendation Layer* is the core part in this framework, which will distinguish it from other distributed data processing frameworks. This layer recommends a suitable approach by predicting the best performance outcome to information processing tasks. The Recommendation Layer is the most important layer because the *prediction strategy* is part of this layer (see Chapter 3).
- *Contributor Layer* provides a series of tools, as well as customised tools, to formalise the types of service contributors, which are based on the features of MDevs.

- *Physical Layer* contains real world users and hardware. Real world users are, for example, data custodians and resource providers. Physical hardware includes smartphones, laptops, desktops, wearable devices and other mobile computing facilities.

The above layers, *Contributor Layer*, *Recommendation Layer*, *Application Layer*

only contain *virtual* service Contributors.

In the *Contributor Layer*, a Contributor Formation Module is used to undertake the transition task from physical devices to different Contributors. This Contributor Formation Module is implemented by each physical device through a pre-installed light widget. The common tools in Contributor Formation Module for Algorithm Contributor/Algorithm Facilitator Contributor, Computing Contributor/Computing Facilitator Contributor and Resource Contributor/Resource Facilitator Contributor Formation include:

- *Contributor Tree Organiser (CTO)*, which maintains and updates different Contributor Trees and handles the requirement of synchronising Contributor Trees with those stored in different Facilitator Contributors.
- *Contributor Generator (CG)*, which initiates a specific type of Contributor. The initialised Contributor contains its Contributor Token, its default state (*Ready* when initialised), and the behavior component which allows the Contributor to communicate with other Contributors.

Apart from those common tools, each type of Contributor also has its specific tools.

When constructing Algorithm Contributor (AC)/Algorithm Facilitator Contributor (AFC), specific tools include:

  - *Algorithm Contributor Activator (ACA)*, which is activated if the physical device applies as a AC.
  - *Algorithm Contributor Tree Organiser (ACTO)*, which controls maintaining and updating the ACT.
  - *Algorithm Contributor Generator (ACG)*, which initiates a unique AC. The initialised AC contains its own ID, comprising the Contributor Type and Time

Stamp, its default state, set as *Ready* when initialised, the component enabling the AC to communicate with other Contributors and other information about the AC's available capability. In addition, one more type of AC is generated by ACG: an Algorithm Facilitator Contributor.

When constructing Computing Contributor (CC)/Computing Facilitator Contributor (CFC), specific tools include:

- *Computing Contributor Activator (CCA)*, which is activated if the physical device will apply as a CC.
- *Computing Contributor Tree Organiser (CCTO)*, which maintains the CCT.
- *Computing Contributor Generator (CCG)*, which initiates a unique CC. The initialised CC contains its own ID, comprising the Contributor Type and Time Stamp, its default state, set as *Ready* when initialised, the component enabling the CC to communicate with other Contributors and other information about the CC's available capability. In addition, one more type of CC is generated by CCG: a Computing Facilitator.

When constructing Resource Contributor (RC)/Resource Facilitator Contributor (RFC), specific tools include:

- *Resource Contributor Activator (RCA)*, which is activated if the physical device will apply as an RC.
- *Resource Contributor Tree Organiser (RCTO)*, which maintains the RCT.
- *Resource Contributor Generator (RCG)*, which initiates a unique RC. The initialised RC contains its own ID, comprising the Contributor Type and Time Stamp, its default state, set as *Ready* when initialised, the component enabling the RC to communicate with other Contributors, and other information about the RC's available capability. In addition, one more type of RC is generated by a RCG: a Resource Facilitator.

When constructing the Computing Facilitator Contributor (CFC), the Processing Request (PR) Generator tool raises a Processing Request, including the properties of the data set to be processed, the resulting format it requires, the time constraints within

which the processing must complete, and privacy concerns.

In the *Recommendation Layer*, one important tool is a *Computing Contributor Recommender (CCR)*, which is used to implement the Prediction Strategy proposed in Chapter 3. Another important tool is an *Algorithm Selection Component*, which recommends suitable algorithms according to the requirements in a Request Message.

The framework over FlexMNet contains Contributors and tools, including one Task Contributor, one Algorithm Facilitator Contributor, one Algorithm Contributor, one Computing Facilitator Contributor, Computing Contributor(s), a Computing Contributor Recommender, Resource Contributor(s) (optional), and an Algorithm Selection Component. All Contributors and tools work in the *Recommendation Layer* and *Data Processing Layer* and *Application Layer* to complete the data processing task.

Each Contributor is equipped with a pre-installed algorithm interface, computing interface or resource interface. These interfaces are only activated if they are assigned to implement tasks by other Contributors. For example, a computation interface in a Computing Contributor will be activated if an Algorithm Contributor chooses it to help organise the Computing Contributor Tree.

The architecture maintains a separation between layers, where a layer serves the layer above it and is served by the layer below it as illustrated in 6.3.

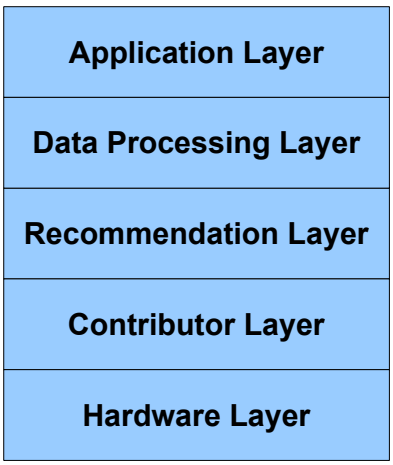

Figure 6.3. Architecture of medium data processing framework over FlexMNet

A summary of the core functionalities of each layer in the framework is presented in

Table 6.2.

Table 6.2. Core functionalities of each layer

Layers	Functions
Application Layer	1) Quick response to external queries; 2) Collaboratively display the result.
Task Processing Layer	Deploy the partitions of the data set to each MDev according to the recommended strategy generated in Recommendation Layer.
Recommendation Layer	1) Recommend suitable approaches by predicting the best performance outcome of data processing tasks; 2) Organise Contributors for Data Processing.
Contributor Layer	1) Designate the physical devices to certain types of service Contributors; 2) Register Contributors in MDevs to FlexMNet
Hardware Layer	Connect available physical devices to a network

6.4 Data Processing Flow over FlexMNet

FlexMNet comprises various types of contributors. Each type of contributor performs different activities when they participate in a medium data processing task. The activities of these Contributors are demonstrated via a general data processing architecture, which is illustrated in Figure 6.4. This architecture highlights how Contributors collaboratively participate in a data processing task, as well as demonstrating the functionality of each layer of the framework.

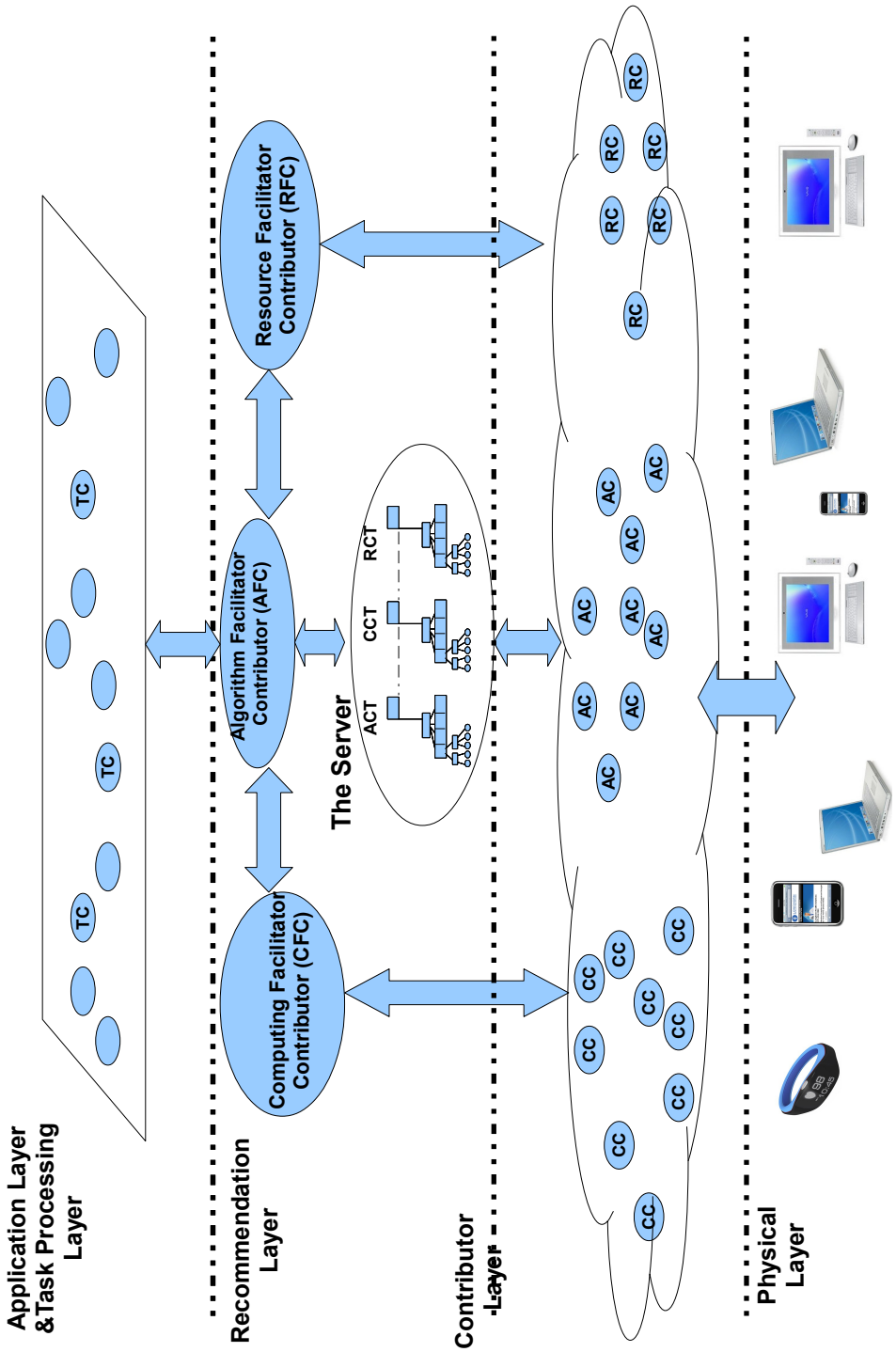

Figure 6.4. Task processing flow of the framework

	TC	AFC	AC (s) *	CCs	Server
INITIALIZATION: A mobile device which owns a dataset initializes a Task Contributor; AFC and CFC have been determined; the Server is online; AFC and data are not in the same device.					
Step 1	Sends the RM to AFC				
Step 2		a.) Broadcasts the RM to all Acs; b.) Contacts the Server to retrieve the available ACs			
Step3			Available ACs notify the Server their availability for the RM		
Step 4					Responses a selected algorithm owned by an AC to AFC
Step 5		Contacts the AC containing the selected algorithm for the RM			
Step 6		Sends the RM and other requirements to CCR			
Step 7					CCR replies to AFC a recommended group and partitions
Step 8			(a) Copies the selected algorithm to each CC in the recommended group		
	(b) Copies the partitions of the dataset to each CC respectively according to the recommended partitions				
				(c) Each CC processes the dataset; (d) Each CC sends the local result to the AFC and then frees the used memory space	
		(e) Merges the result as a final result			
Step 9		a) Returns the result to the TC; b) Releases the AC and CCs			
END: Terminates the RM after receiving the satisfied result					
* From Step 2-4: all alailable ACs; From Step 5: a specific AC					

Figure 6.5. An instance of data processing over FlexMNet

The following steps summarise the activities of these Contributors during one session of processing a medium data set in a distributed manner over the FlexMNet, see Figure 6.5. The initialisation of the process includes:

i) The assumption that an Algorithm Contributor (AC) has successfully bid to be an Algorithm Facilitator Contributor (AFC);

ii) The supposition that an AFC in this session undertakes the duties of a Computing Facilitator Contributor (CFC);

iii) A MDev initialises a Task Contributor (TC) to process a data set which stores in it.

DO FOREVER

(**Step 1**) *Task Contributor (TC)* launches a Request Message (RM) containing Processing Request to the Algorithm Facilitator Contributor (AFC);

On receipt of a RM, AFC does:

(**Step 2**) AFC broadcasts the RM to all Algorithm Contributors (ACs) and

(**Step 2**) AFC communicates with the Server to retrieve the available ACs.

Then,

(**Step 3**) Available ACs notify the Server of their availability for the RM;

(**Step 4**) The Server selects an algorithm satisfying the requirements of the RM by searching Algorithm Contributor Tree (ACT), then sends the Token of a selected algorithm owned by an available AC to the AFC, and meanwhile sets the State of the available AC to *Occupied* ;

(**Step 5**) AFC contacts the AC containing the selected algorithm for the RM;

(**Step 6**) AFC sends the RM and other requirements (e.g. privacy) to Computing Contributor Recommender (CCR) located in the Server;

(**Step 7**) CCR sends a message to AFC containing the recommended group members and how to partition the data set;

(**Step 8**) AFC sets the State of the Computing Contributors (CCs) in the recommended group to *Occupied* then:

(**Step 8.a**) Asks the AC to copy the selected algorithm to each CC in the recommended group;

(**Step 8.b**) Asks the DO to copy partitions of the data set to each CC respectively in the group;

(**Step 8.c**) Each CC starts processing the data set;

(**Step 8.d**) Each CC sends the local result to the AFC and then frees all temporary information for this computing in the memory space;

(**Step 8.e**) Collects the local results from each CC and merges the results as a final result;

(**Step 9.a**) AFC returns the result to the TC;

(**Step 9.b**) AFC sets the State of the available AC back to *Ready* in the ACT in the Server and releases all CCs;

(**END**) TC terminates the RM when it receives a satisfied result from AFC, or otherwise starts another RM as in Step 1)

END

The framework starts from the time point that a Task Contributor throws a Processing Request and finishes at the time point that the final result for this Processing Request is published. The process described above is an instance of this framework. A real world application that uses this framework to process their data may contain a number of this instance, which need to be implemented in parallel by different Contributors.

6.5 Case Study: Data Processing in a Medical Environment

Digital wearable devices and other portable MDevs have recently become widely used in collecting a patient's medical data, such as heartbeat, skin temperature, sleep/wake patterns, and activity levels depending on the requirements of medical treatments. In

2016, these devices are not only able to collect data, but also have much more computing capability to process a large amount of data compared to the devices twenty years earlier. Wearable and MDevs show, for example, the possibility for monitoring a number of medical risk factors and giving patients direct access to their personal analytics that can benefit their health, contribute their preventive care, and aid in controlling their ongoing illness (Piwek, Ellis, Andrews & Joinson 2016). Furthermore, these devices assist healthcare professionals, such as doctors and nurses by monitoring, collecting relevant health data of healthcare consumers and analysing them in real time. Additionally, the patients or healthcare consumers can process the collected data themselves via real-time analysis tools/apps to understand their medical conditions.

Another application of these devices is to assist health and life sciences companies to conduct clinical trials process. The clinical trials process can be optimised by leveraging existing smart technology (Patient Empowerment Network 2015). For example, smart pill technology (also known as *ingestibles*) which allows for both wireless patient monitoring and diagnostic imaging. (If health and life science companies can get enough insight in the early stage of development, they can potentially create a more efficient drug development process and prioritise resources for the most promising therapies, with the goal of getting effective drugs to market faster.) When wearable technology is applied to clinical trials, it becomes a potentially powerful research tool for remotely monitoring and gathering clinical data of patients in real-time.

These applications often need wearable and MDevs to process the data collaboratively, especially when the processing is computing-intensive. For example, some medical data comprises multiple dimensions and are difficult for its DO to process. Some medical data are collected from intensive data streams, which result in a comparatively large volumes of data within half a day. A cooperative Flexible-sized Mobile devices Network (FlexMNet) is designed for processing this computing-intensive data set within a limited time framework by optimising the possible computing resources nearby. A case in which a patient's healthcare data is processed over FlexMNet is described as follows.

In a department of a hospital, the Internet connection is disabled because of privacy

concerns and also for security reasons around medical devices. However, the hospital is covered by a local network, such as Wi-Fi. This FlexMNet comprises staff members working in this hospital and patients who have treatments and personal information in this hospital. Therefore, devices which register as Contributors in this network have been authorised to access and retrieve personal information stored in other participating devices. A Server of FlexMNet is installed within the local network. The Server has sufficient computing capability to store the registration information of the FlexMNet members, as well as the organisation information of resources for processing data, such as algorithms, data sets and computing devices.

Users register themselves as a member of a FlexMNet created and managed by a department in hospital. Accordingly, patients or doctors or nurses or another staff members working in the hospital can registers themselves as various Contributors depending on what resources they are willing to contribute. For example, patients or healthcare consumers register as Computing Contributors or Resource Contributors while medical professional staff members register as Computing Contributors or Algorithm Contributors. The interface is an registration mobile applications to the FlexMNet which is able to be installed in each MDev.

A patient in a department has a wearable device that monitors his/her healthcare conditions. The data collected by the device contains several dimensions, heartbeat, skin temperature and other activity levels. The data produced by the wearable device has to be sent to the company that produced the device to get specially analysed information. The patient is able to receive information about his/her medical conditions from the specially analysed information produced by the company every half a day.

Additionally, he/she wants to know more than the company has provided about his/her data. So he/she has the following requirements;

- the results from the data as a form of clusters based on heartbeat, skin temperature and EEG (electroencephalogram) results; and
- the results needs to be returned to him within half an hour after his/her request has been sent.

However, his/her wearable device is not able to process the data stored in it due to two

restrictions;

- the computing capability of this device is not powerful enough to process the data; and
- the device does not have the algorithm installed to process the data.

Therefore, he/she has to ask for an assistance of the FlexMNet in the hospital to process his/her personal request. The following steps explains the procedure in which the patient's data is processed by applying the framework over the FlexMNet;

i) The patient operates his device to register a Task Contributor to the Server;

ii) The Server arranges an AFC to his/her device which is also agreed upon and trusted by the patient (if the patient does not have the computing capacity to deal with the workload of AFC);

iii) The patient sends his/her data processing request within the Task Contributor to the AFC;

iv) The Server replies an appropriate AC;

v) The AFC contacts the AC to confirm the the availability of the AC;

vi) The AFC broadcasts the request to the available CCs geographically located in the department building to get assistance;

vii) Several CCs reply their availability to the AFC. Now a medium data processing framework over FlexMNet is constructed;

viii) A group of available CCs are recommended by the Computing Facilitator Model;

ix) In the next few steps, the group of CCs are organised to collaboratively process the data and generate a result. Refer to Step 5 in Figure 6.5 for the details of the rest steps.

x) Return the result to the patient.

6.6 Summary

The requirement of appropriate organisation motivates the work presented in this chapter. A medium data processing framework of a FlexMNet is proposed. The purpose of this framework is to process a computing-intensive data set within a time constraint and by optimising the possible nearby computing resources with a better performance. This framework simulates computing techniques found in *distributed and parallel computing* and behaviors of intelligence in *agency-based and mobile-based DM* and *social group work.*

A FlexMNet is constructed from a number of synergetic mobiles devices which do not necessarily have the same computational capability or resources. These devices must first be voluntarily registered as being available for data processing tasks. The topology of the network is dynamic and self-configuring, based on the specific task requirements, available devices and the properties of the devices themselves. The performance of a data processing task for *medium data* within a FlexMNet is degraded if the data set has not been split to appropriate partitions suitable for each MDev. Alternatively, the performance of the task can be affected if the available MDevs in FlexMNet are not organised optimally.

The inspiration for the framework over FlexMNet is from the assumption that *a collaborative group can finish a particular workload in the most efficient way with a minimum privacy leakage if the distribution strategy is properly chosen.*

This chapter also presented a possible application of this Framework. There are still considerable details that need to be discussed to implement this application, such as cooperation with various MDevs, integration or communication of different local networks and communication and security protocols. Implementation of this framework is beyond the scope of this research, which however can be the future work of this research.

Chapter 7

Privacy Protection Over the FlexMNet

7.1 Introduction: Privacy Issues in the FlexMNet Framework

When processing data sets in any ad hoc mobile network, security and privacy are two of the major issues.

Various security issues include: availability, data/message integrity, confidentiality of certain information, authenticity and authorisation. However, because privacy is the major concern in the FlexMNet Framework, the issues regarding security are beyond the scope of this research.

The main privacy issues that need to be considered after proposing the medium data processing framework over FlexMNet are:

i) The data processing tasks are ad hoc and provisional, thus applying complicated Privacy Preservation (PP) strategies are not necessary. However, some devices participating in tasks have some privacy concerns, although not a significant amount.

ii) Participating devices do not have a significant amount of privacy that needs to be preserved. However, applying traditional complicated PP algorithms in their data sets is still difficult for those devices due to limited computing capabilities caused by either slow processors or limited storage space.

This chapter will discuss privacy issues arising in the proposed FlexMNets Framework. Recall the three typical scenarios for the Framework presented in Chapter 6:

- Scenario 1: Participants who trust each other form a collaborative data processing network, in the situation where processing the data set is much more important than protecting personal privacy in individual devices. An example of this is that sensor networks are used to collect the real-time weather data deployed by the same institute.

- Scenario 2: Participants who do not trust each other form a collaborative and temperate processing network in order to complete a single task, in which the information to be processed is public. The task must normally be completed in a short time, e.g. less than one hour. Once the task is completed, the network is not valid. For example, a foreign visitor needs a translation of a long paragraph, explaining the art that he/she is viewing while in a museum. The translation task is beyond what his/her MDev can process, thus he/she needs the help of other visitors who have MDevs, in order to form a temporary network. This translation task must be completed in a few minutes because conventionally a museum visitor would not like to spend a few hours on one piece of art due to time constraints.
- Scenario 3: Participants who do not trust each other form a collaborative data processing network to temporarily complete a single task, where the information to be processed is private. The major differences in this scenario compared with the second one are; 1) a larger amount of information; 2) the shared information is private; 3) the task must be completed within a certain time frame, but uses a longer processing time; and 4) as the task asks for a longer processing time, the temporary network requires better stability.

There is no privacy concern for the shared data set in Scenario 1 as the data to be shared is in the same schema or is managed under the same authorised body. Although Scenario 2 is not a trusted network, the shared information is public, and no participant's privacy is involved. In this scenario there are security issues in participating MDevs, which is, however, beyond the scope of this chapter. Scenario 3 works for the mobile DOs who have sensitive data sets to process but do not have the capability to process the data sets and to apply comprehensive privacy protection strategies. In this situation, privacy protection constraints need to be applied in this scenario. As a result, the privacy protection issues in Scenario 3 is the focus of this chapter.

In terms of the major features of MDevs used in the framework over FlexMNet, such as small capacity, the privacy protection issue that this framework is particularly concerned with is how to reduce the amount of sensitive information for a computing task.

This chapter discusses this point from two perspectives: identifying high risk sensitive data; and finding the most cost-effective privacy protection strategies.

Identifying high risk sensitive data has been widely discussed in the field of *risk management*. In risk management, identifying an asset, i.e. what is worth protecting, is always applied as the first step.

When risk management is planned for a task, the most important step is to define the assets. In the framework over FlexMNet, data is always the major concern. In Information Security, there is a common concept, which also causes a dilemma; the more security that is applied to the asset, the less liberties are available to the users. Accordingly, an appropriate amount of assets needs to be carefully determined for any particular case. A similar concept is also applicable in sensitive data protection: the more data is protected, the less likelihood there is that the data is acquired by malicious intruders, and there are fewer malicious actions that those intruders can apply to the data set. However, there would be less usability/functionality that appropriate users can apply to that data, and more cost would be involved. For this reason, compromise and discussion-based strategies between the DO and protection strategy provider should be considered. In Risk Management, those strategies proposed in the step of risk mitigation development, are conducted by services such as discussions/ meetings/ reviews (Whitman & Mattord 2016). These services should also apply to personal users such as the ones who form a temporary computing network, FlexMNet. Digital tools can provide these services to personal mobile users to assist them to self-identify their high-risk data and to determine a solution to protect their sensitive data. Because of the financial restrictions, these tools need be designed with a few features, and they should be convenient, cost effective and embedded in the users' MDevs.

Therefore, this research studies computer-based strategies for conducting discussion or meeting services. These strategies are useful for individual DOs who do not have essential financial support to protect their data. These strategies will also benefit organisational DOs who need to balance the performance of utilising their data with the financial expenses associated with protecting their sensitive data.

In summary, considering the issues addressed in Scenario 3) for DOs with limited

processing resources, it is essential that two tasks are covered in this research:

Task (A) identifying high-risk sensitive data in a shared data set; and

Task (B) designing lightweight, convenient privacy protection algorithms.

The rest of this chapter discusses the issues involved in tasks (A) and (B) and proposes possible strategies for each task.

7.2 Task (A): Identifying High-risk Sensitive Data in a Shared Data Set

For any data processing system including a set of hardware, software, data, people, procedures and networks, the combination of these components gives rise to an enterprise wide resource that becomes a business enabler in contemporary enterprise environments. From a technical perspective, the major focus points are centred around the technological aspects, particularly the hardware, software, data, and networks. As the major functionality of the FlexMNet Framework is data processing, data is the major asset when sensitive data protection is concerned. Privacy protection of data is concerned with its availability and accessibility in conjunction with its overall integrity. Often considered as the most valuable asset, and therefore the most highly prized target, the data belonging to a DO must be protected during its lifetime, from acquisition through to transmission and processing, and finally during its storage.

7.2.1 What Sensitive Data Stored in a MDev Does the DO Need to Protect?

Multiple nodes in the network are involved in processing data, and the relaying of packets/data has to be authenticated by recognising the DO of the packet/data and de-identifying the data or label and their locations. In most cases, DOs' identifications and locations will not affect the result of data processing. In consequence, this information is identified as high-risk sensitive data, in order to avoid unnecessary privacy leakage.

Apart from these two widely accepted types, what other types of sensitive information must be protected? How can a DO recognise them and understand their potential risks? Answers to these questions are explored by analysing the following cases suitable to the FlexMNet Framework.

- Case 1: Wearable devices, also known as wearable technology, are widely accepted by the public to improve healthcare well-being. They provide convenience and accurate medical data that can benefit a doctor's decision making process. For example, a wearable tracking device, named Thim, re-trains insomniacs to get a better night's sleep at home. This device connects wirelessly to a smartphone app to determine when the person is awake or asleep (Lack, Scott, Micic & Lovato 2017). However, personal information such as patients' identities and real-time locations stored in medical data are always sensitive. Therefore, personal information which need to be hidden from third parties analysing the medical data .
- Case 2: Collaborative data analysis is conducted among several DOs, who are also competitors in the market. In this case, collaborative data analysis is essential among the DOs. However, each DO worries about the malicious leakage of important information to competitors or any unintended actions conducted by their data share partner. For example, two retailers that sell complementary products are interested in sharing data to identify and exploit potential cooperation that would enable increased profits for both of them. However, some data stored in their databases might be critical to their business strategies and must not be revealed to their competitors. Cooperation with their peer retailer will increase the risk of leaking the critical information to a competitor (Menon, Sarkar & Mukherjee 2005).
- Case 3: A retailer is implementing several shelf-space allocation strategies. These self developed strategies currently resulting in very profitable market baskets being purchased in one of the retailer's shop. However, he/she worries that if the sales data are provided to a third party for further market analysis, disclosing these market baskets results in risks of their competitors replicating

these strategies (Menon et al. 2005). In (Verykios, Bertino, Fovino, Provenza, Saygin & Theodoridis 2004, Oliveira & Zaïane 2003, Evfimievski, Gehrke & Srikant 2003), other examples of situations are discussed where the sharing of unaltered databases can have serious adverse effects.

In Case 1, medical information collected by wearable devices is necessary in order to analyse the sleeping status of a patient. However, the patient's identity and real-time locations will not affect the result of the medical data analysis. For this reason, only the patient's identity and locations need to be hidden from the data set analyser.

In Case 2, the sensitive data to other business competitors can be the number of products sold in each single transaction and the store locations that participate in the collaborative data processing task.

In Case 3, the sensitive data that need to be protected are the number of products sold and total amount in each single transaction. As a result, there is no need to apply the privacy protection strategy to the entire data set.

Therefore, a quick and interactive privacy preservation strategy between DOs and protection strategy providers is required to address these above situations.

Most privacy protection algorithms are not suitable for MDevs because of their complexity and large computing cost.

Because of the limited capacity of MDevs, determining the minimum amount of privacy assets is critical. In the FlexMNet Framework, the task of identifying the assets is assigned to MDev owners as I believe it is the DOs who understand their privacy concerns the best. This design can assist DOs to determine their privacy assets as accurately as possible.

7.2.2 Bands of Privacy Protection Assets

As discussed in the previous section, for a temporary data processing task in the FlexMNet framework, determining the minimum amount of privacy asset, which is high-risk sensitive data, is important for improving the performance of processing the task. Therefore, **Bands of Privacy Protection Asset (BPPA)** 7.1, the basic concept of

which has been published in (Li, de Vries & Roddick 2011), is proposed in this section to assist MDev owners to identify their minimum privacy assets in the framework.

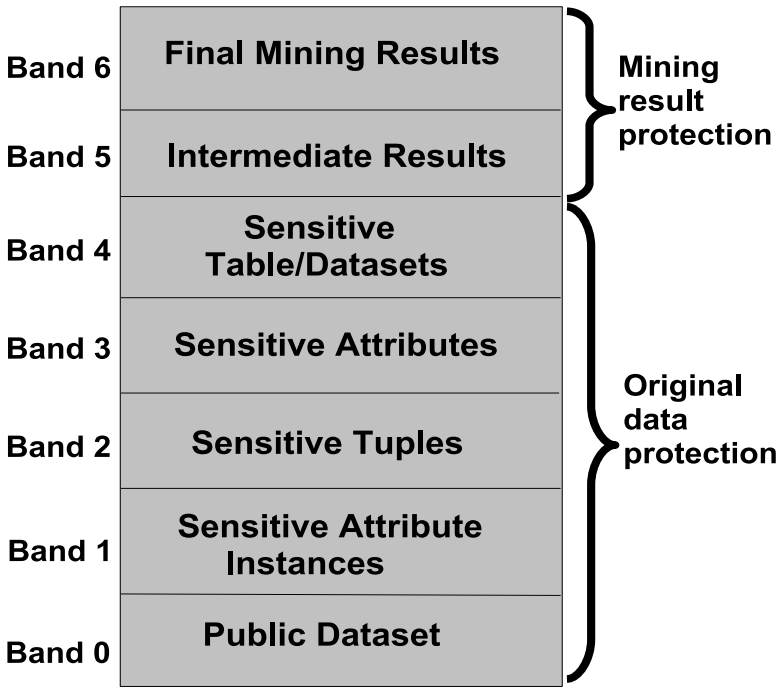

Figure 7.1. Bands of Privacy Protection Asset

The BPPA is a technology applied in the privacy protection component in the FlexM-Net Framework as it provides a quick service for the DO. Each band gives a definition of the information which needs to be protected, which helps DOs to identify their assets for privacy.

Most of the time, the security of a data processing task implemented through the FlexMNet framework is not rigid. Therefore, a table with two dimensions is provided for privacy protection for both DOs and service assistants. DOs choose which level of privacy protection defined by the BPPA they need to achieve, and service assistants provide the privacy protection strategy that DOs can conduct by themselves. The intersection Tuple indicates the actual strategy which would be able to be implemented for DOs.

A thorough analysis of privacy requirements reveal the BPPA presented in Figure 7.1. The BPPA have two groups: 1) data protection for original data sets, including Sensitive Attribute Instances, Sensitive Attributes, Tuples, and data sets; and 2) protection

for the information from the mining results, including Intermediate Mining Results and Final Mining Results. This classification of PPDM enables domain experts to more easily select the appropriate DM algorithm based on the nature of the data set.

The following bands classify the assets of privacy protection for PPDM algorithms.

Band 0: The Public data set

Centralised DM algorithms which are not concerned with privacy issues fall into this band.

Band 1: Sensitive Attribute Instances Protection

PPDM algorithms should protect the sensitive values of attributes in particular cells. Normal privacy preserving techniques, i.e. modifying data, deleting data and swapping data, can be adopted to sanitise these values. An example case is as follows:

> An institute owns a data set, consisting of the environmental quality factors in horizontal attributes and the dates within a month in columns. The values above a particular threshold value in a certain attribute column are confidential. Therefore, before performing the DM process over this data set, these confidential values should be protected through some privacy protection strategy.

Band 2: Sensitive Tuples Protection

The protected objectives in this band are sensitive records or data in row. In circumstances where the number of records and the names of attributes are not deemed as sensitive information by DOs, the PPDM strategies applied on a horizontally partitioned data set are considerable. Typically, randomisation techniques, distributed data set techniques and mathematical transformation approaches are used to sanitise data in

this band. However, geometric transformation strategies do not apply to privacy protection in this band because of the diversity of data types in a record. An example of this is as follows:

> A medical company is obliged to stop any trace of individual profiles of the patients who have *lung cancer*. This obligation requires the hiding of all records for these patients when the value in the *disease* attribute is equal to *lung cancer*.

Band 3: Sensitive Attributes Protection

The protection objective in this band is the sensitive column or attribute. In multi-party schemes, when the data set is vertically partitioned over several DOs, DOs are willing to protect whole sensitive attributes they hold. An example case (Yang & Huang 2007) is:

> A commercial bank has its record of customers including attributes for date of birth, income, deposit and credit. It is possible for adversaries to obtain a sort on attributes in descending order. It is also possible for adversaries to acquire valuable information by launching aggregated queries, e.g. SUM, MAX. The leakage risk should be eliminated by protecting the sensitive column. As there is the same data type for the entire column, most traditional geometric transformation methods are good solutions for protecting objectives fitting this band.

Band 4: Sensitive Table or data set Protection

In this band the whole table or whole data set requires protection. As the privacy protection tasks on this band normally involve a distributed data set, the PPDM algorithms using only modification or transformation techniques do not satisfy the privacy preservation requirements. Thus, from this band to Band 6, Secure Multi-party Computation

(SMC)-based and query-based PPDM algorithms are introduced. In some cases, to protect the sensitive information in this band, the partition of the data set can be very complex. For example, the instances for one attribute can be held by several DOs, A, B and C, while the instances for another attribute are held by DOs, D, E and F. Namely, *no* DO holds *all* instances for *one* attribute and *no* DO holds an entire record with values for *all* attributes.

An example case is as follows:

> A company has a data set storing 10 years of financial information. For security reasons, the data set is shared between different departments, A, B and C. Each department only holds part of the data set, so that no department knows all attributes of the data set nor the total number of the entire records. The company is interested in exploring new knowledge by mining over their entire data set, but with the constraint of no disclosure of the data set to other departments and the external DM company.

The data set D (Figure 7.2) is partitioned into four parts, D_A, D_B, D_{C1} and D_{C2}, owned by three DOs A, B and C respectively.

> D has n attributes $\{A_1, A_2, \ldots, A_n\}$ with a total of m records $\{R_1, R_2, \ldots, R_m\}$. After partitioning, A gets data set D_A comprising s records $\{R_1, \ldots, R_{1+s}\}$ of t attributes $\{A_{n-t}, \ldots, A_n\}$; B gets data set D_B comprising u records $\{R_{m-u}, \ldots, R_m\}$ of n-t attributes$\{A_1, \ldots, A_{n-t}\}$; C gets data set D_{C1} comprising m-u records $\{R_1, \ldots, R_{m-u}\}$ of n-t attributes$\{A_1, \ldots, A_{n-t}\}$ and data set D_{C2} comprising m-s records $\{R_{m-s}, \ldots, R_m\}$ of t attributes$\{A_{n-t}, \ldots, A_t\}$. The DM tasks will be applied over the union of data sets from the three DOs, while each data set, D_A, D_B and (D_{C1}+D_{C1}) must be kept confidential from the other two DOs.

There are two options for performing the mining tasks: providing the DM company with a modified data set or implementing a SMC protocol. However, as the records of

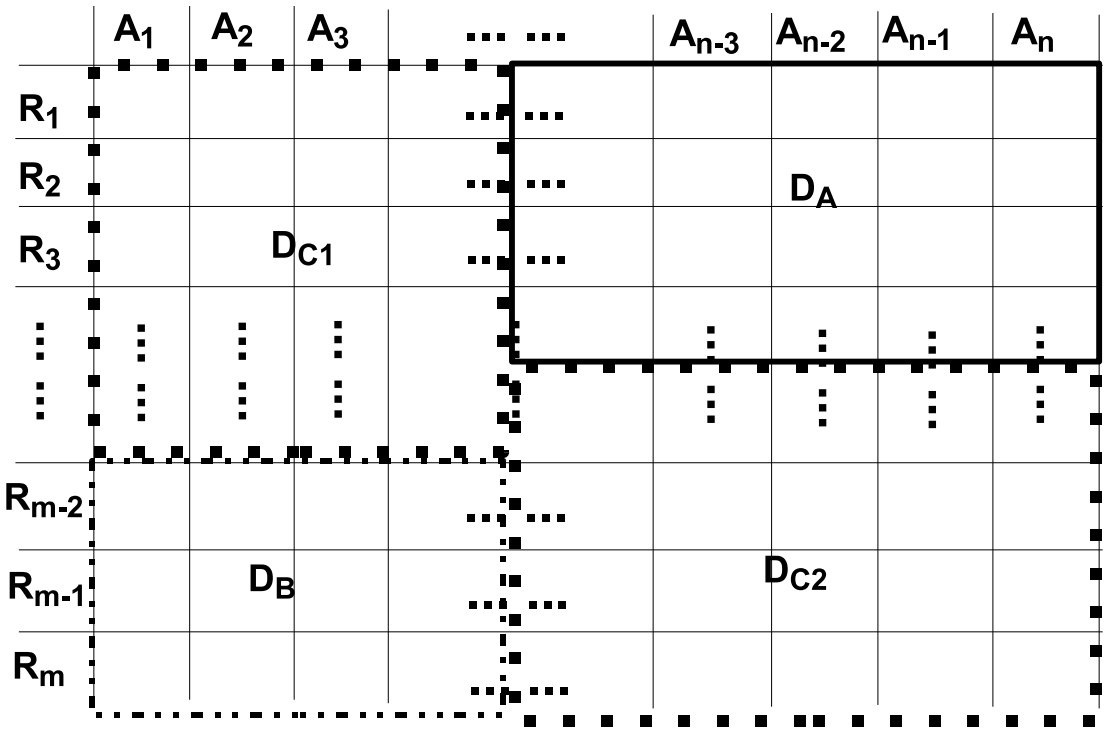

Figure 7.2. Partition of a united data set in Band 4 protection

one attribute are owned by more than one department, the attribute protection strategies cannot be adopted in this case. Likewise, the records with all attributes in the united data set are managed by more than one department. The partition of one record restrains the application of the tuple protection strategies. Hence, cases in this band are more complicated than in bands 2 and 3. The cases in this band are not able to be dealt with by the strategies for horizontally partitioned or vertically partitioned data sets.

Band 5: Intermediate Result Protection

The protection objectives in this band are the sensitive intermediate results, that is, the frequent itemset from the association rule mining or the intermediate clustering core points from clustering. We separate this band from the Band 6 as, in some cases, the intermediate results are sufficient for adversaries to trace related private data.

Band 6: Final Mining Result Protection

The aim in this band is to protect the rules or clusters generated from the private data sets. Here the information that should be prevented from disclosure is:

Any clusters or patterns from the sensitive data sets which may lead to sensitive information.

Using mining results with either public or private data inferences may be made to identify individuals.

A widely cited study by (Sweeney 2000) and revisited by (Golle 2006) showed how using publicly available information (sex, ZIP code and date of birth) with census data allows for the unique identification of individuals. Calandrino et al. (2011) used a moderate amount of auxiliary information about a customer and inferred this customer's transactions from temporal changes in the public outputs of a recommender system.

7.2.3 Using BPPA in the FlexMNet Framework

BPPA is applied in the Algorithm Selection Component in the framework over FlexMNets to help DOs determine the minimum amount of essential sensitive information in the DOs shared data set. By clarifying the sensitive data, computing resources can be saved to concentrate on the actual computing task in a situation with limited computing resources.

The Process of applying the BPPA in the framework over FlexMNets is illustrated in Figure 7.3.

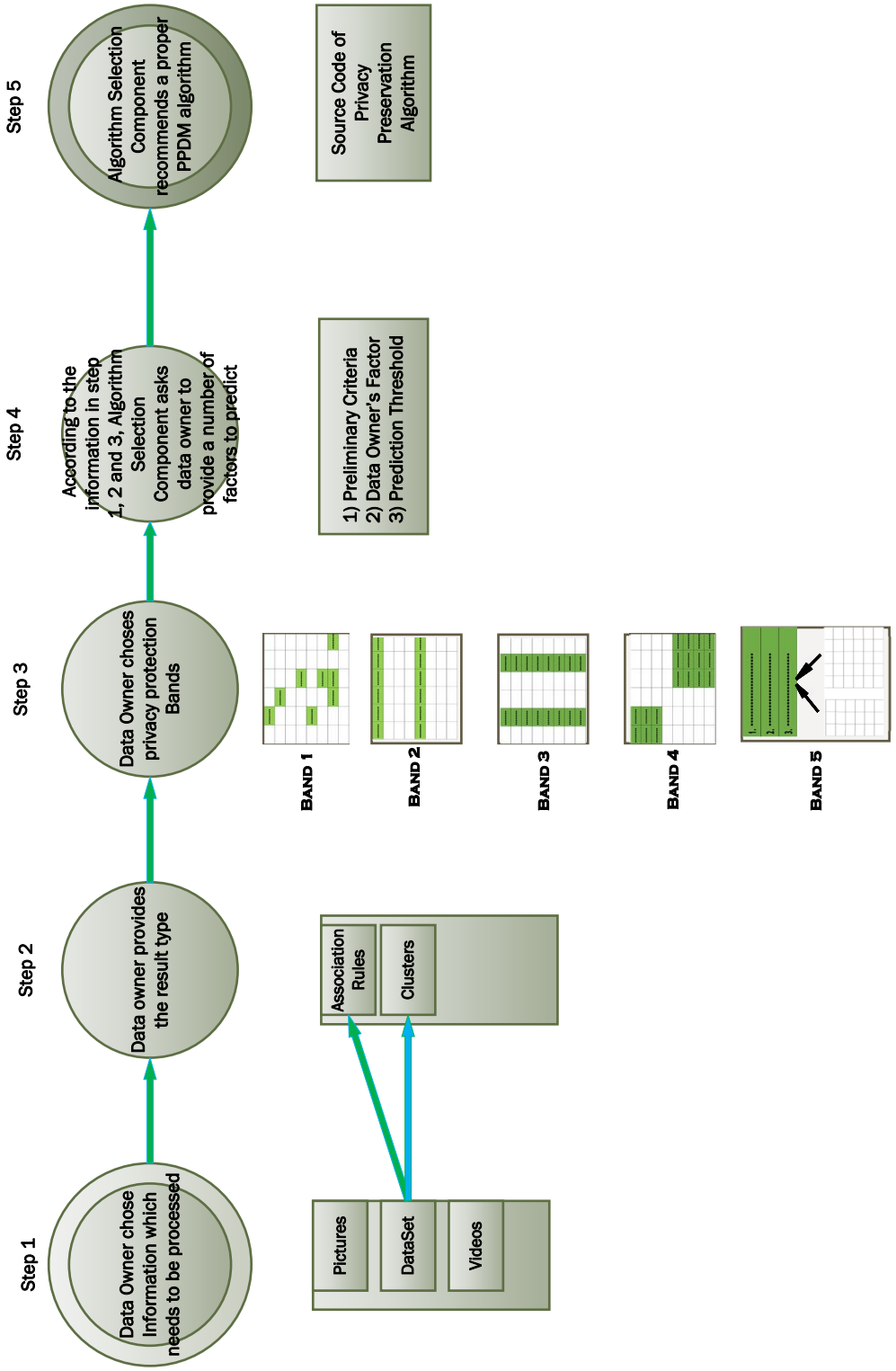

Figure 7.3. Process of applying BPPA in the framework over FlexMNet

When a DO has a request to process a data set, the Algorithm Selection Component (described in Chapter 6) recommends an appropriate data processing algorithm. If

a DO has privacy concerns in the data sets, a privacy preservation algorithm which is suitable for both the data processing algorithm and DOs privacy concerns will be applied. The privacy preservation strategies are recommended. The functionality of recommendation can be undertaken by an additional tool in the FlexMNet framework, the Privacy Preservation Recommendation Component. A detailed process is also described in Figure 7.3 with six steps:

Step (1): DOs choose information which needs to be processed;

Step (2): DOs provide result patterns that they are expecting to view, e.g. Association Rules, Clusters;

Step (3): DOs choose one band of privacy protection from the list;

Step (4): In terms of the information collected from Step (1), (2) and (3), the Algorithm Selection Component asks DOs to provide the information for a number of Data Owner's Factors (DOF), explained in the next section, for predication in Step (5);

Step (5): The Algorithm Selection Component recommends a proper privacy preservation algorithm from a number of intentional algorithms for DOs to apply on their data sets after considering the information collected from the steps above. A prediction strategy, explained in the next section, is adopted in order to consider the information;

Step (6): DOs copy source code of the recommended PPDM algorithm into their cache, and then start applying the code on the data sets.

7.3 Task (B) : Designing a Light Weight and Convenient Privacy Protection Algorithm

Another task for improving the performance of data processing in the FlexMNet framework from the perspective of privacy protection is to design light-weight Privacy Protection (PP) algorithms. Appropriate algorithms suitable for this framework must be

designed to meet a series of criteria and pass a threshold based on a prediction strategy provided by the framework. A series of preliminary criteria and DO's Factors are discussed in the next section. DO's Factors are considered in the prediction in order to select a proper PP algorithm.

The aim of a privacy protection strategy used in FlexMNet framework is to protect the privacy of data which is 1) stored in MDevs; and 2) to be shared within FlexMNet when undertaking a data processing task. Security issues regarding the information stored in MDevs are beyond the scope of this chapter.

However most of the prevailing MDevs have very limited storage capacity. For example, as released to the market in 2016, a smartphone has a storage capacity from 16 to 128 GigaBytes and a laptop with wireless adapter has a storage capacity of up to 1 TeraByte.

In FlexMNet framework, considering the complexity of a PP algorithm is critical. There are two types of complexity for an algorithm, time complexity and space complexity. In the framework, space complexity must be considered with higher priority than time complexity, as running out of space in a MDev can directly result in the failure of the algorithm. Accordingly, the following considerations regarding preliminary criteria and factors for prediction are mainly based on space complexity and time complexity.

7.3.1 Preliminary Criteria

According to the features of tasks processed by the FlexMNet framework, the following preliminary factors must be considered when selecting a number of candidate PP algorithms from the intentional algorithms before starting the prediction procedure:

- the source code of the PP algorithm must be supported by the MDevs;
- the size of the source code should not take much transmission time and space in DOs' devices;
- safety ranks of the PP algorithm provided by other users must satisfy the safety rank required by DOs; and

- the PP algorithm must provide functions to manage resources in order to minimise the usage in memory space. An example of this is the function that overrides previous intermediate results/files with new intermediate results/files if overriding the previous ones would not affect the generation of the new ones. Other examples of functions can be the cleaning up of temporary files and eliminating memory leaks.

PP algorithms meeting the detailed requirements of these preliminary criteria become the candidate algorithms, which are passed to the next stage, prediction.

7.3.2 Prediction

To satisfy DOs' requirements from the perspective of both privacy protection and their limited/constrained computing resources, DOs are asked to provide a number of Data Owner's Factors that comprise of a number of thresholds which the candidate PP algorithms must satisfy. Those factors are used in prediction.

- Space Prediction estimates the maximum storage space used while applying the PP algorithm, according to the complexity of the algorithm.
- Time Prediction estimates the maximum time taken while applying the PP algorithm, according to the complexity of the algorithm.

Traditional strategies for quantifying the complexity of an algorithm in computer science are adopted to predict the space and time complexity of a privacy protection algorithm. For example, the approach that counts the number of elementary operations performed by the algorithm (Goel 2012) is adopted in the prediction strategy . However, as the size of data is known and the algorithm is implemented in a real situation, the Big O notation (O), which is commonly used to represent the complexity of an algorithm in computer science, is not used to present the result in the prediction procedure. Instead, an expression which sums the execution times of each elementary operations and provides a better precision for the complexity of an algorithm is required.

Take an example of calculating the product of two matrices (M and N) with n dimen-

sions $P = M \times N$ where both M and N are $n \times n$ matrices. The pseudocode of source code and the times that each code is executed are presented in Table 7.1.

MatrixMultiply Function:

Initialisation: int $M_{[n][n]}$, int $N_{[n][n]}$, int $P_{[n][n]}$; int i ,j ,k;

Pseudocode of source code	Execution Time
(1) for (i=0; i¡n; i++) {	*n+1*
(2) for (j=0; j ¡ n; j++) {	*n*(n+1)*
(3) $P_{[i][j]} = 0;$	n^2
(4) for (k=0; k ¡ n; k++) {	$n^2 * (n+1)$
(5) $P_{[i][j]} += M_{[i][k]} * N_{[k][j]};$	n^3
(6) }	N/A
(7) }	N/A
(8) }	N/A

Table 7.1. Execution time of a matrix multiply function

The information in italics after each clause is the number of each clause. The time taken for this algorithm in total is

$T(n) = 2 * n^3 + 3 * n^2 + 2 * n + 1.$

Thresholds refers to a value set to filter out those inappropriate PPDM algorithms. They are also user-defined factors. Only the algorithms passing these user-defined thresholds are recommended to DOs.

7.4 Summary

Typically, PPDM approaches are categorised by the strategy they take or the particular type of DM to which they apply. The ambiguous definition of the *protected objectives*

of a privacy preserving strategy reveals issues from three aspects: the assessment criteria of PPDM algorithms; the DOs' demands; and the balance between the computation cost and privacy leakage.

This chapter presents six bands of privacy preserving objectives for a reconceptualisation of PPDM algorithms. As an initial work, selected existing privacy preserving algorithms have demonstrated the possible use of the BPPA. This Band provides an opportunity for the FlexMNet framework to improve its capability in protecting sensitive data for DOs. Additionally, a procedure of applying BPPA in the framework of FlexMNet is presented.

However, apart from the time restriction of completing this thesis, this thesis will not cover those works for several reasons. First, this thesis mainly concentrates on improving the performance of the FlexMNet framework. Sensitive information protection is one functionality of the framework, rather than the major functionality of this framework. Secondly, the Band is an optional technology for the framework in order to determine a minimum amount of sensitive information for a computing task. This Band can be replaced by other technologies to protect sensitive information which can provide better outcomes for adaption in an instantiated system by the framework.

This chapter also proposes a number of criteria and a prediction strategy with a number of factors to be considered. These criteria and the prediction strategy are used to assist DOs to select the most suitable PP algorithm to apply to their data set. Furthermore, these criteria are discussed in terms of the issues presented or discussed in a number of publications.

Chapter 8

Conclusion and Future Work

Traditionally, computing-intensive tasks have been processed by powerful computing device(s), such as supercomputers and computing grids (reviewed in Chapter 2). Mobile devices with reasonable capability of running mobile Apps and processing information are currently prevalent. This thesis discusses the possibility of harnessing the combined resources of a number of mobile devices with limited computing capabilities to complete computing-intensive tasks, especially in scenarios which lack powerful computing devices.

The network of mobile devices which is organised to collaboratively process computing-intensive tasks is named in this thesis as Distributed Adaptive Data Analysis Network of Flexible-sized Mobile Devices (FlexMNet). By reviewing previous work, this thesis realizes that the performance of processing computing-intensive tasks over a FlexMNet is the major hinderance to the processing from being applicable. This thesis discusses a number of strategies to improve the performance of the processing from different areas, including computing workload allocation, algorithm determination, mobile device organization and sensitive information protection.

Figure 8.1 summarises the contribution of this research by giving an overview of the strategies proposed for improving the data processing performance for a network of mobile devices.

Chapter 3 describes the proposed prediction strategy for improving the performance of a distributed data processing algorithm over the FlexMNet. The strategy comprises three major steps, illustrated in Figure 8.1: predicting the partition of a data set for selecting an appropriate processing algorithm in Chapter 4; participating mobile devices selection in Chapter 3; and hiding a reasonable amount of sensitive information if data owners have privacy concerns in Chapter 7. An over-arching framework is proposed comprising these three steps in order to process medium data over a FlexMNet.

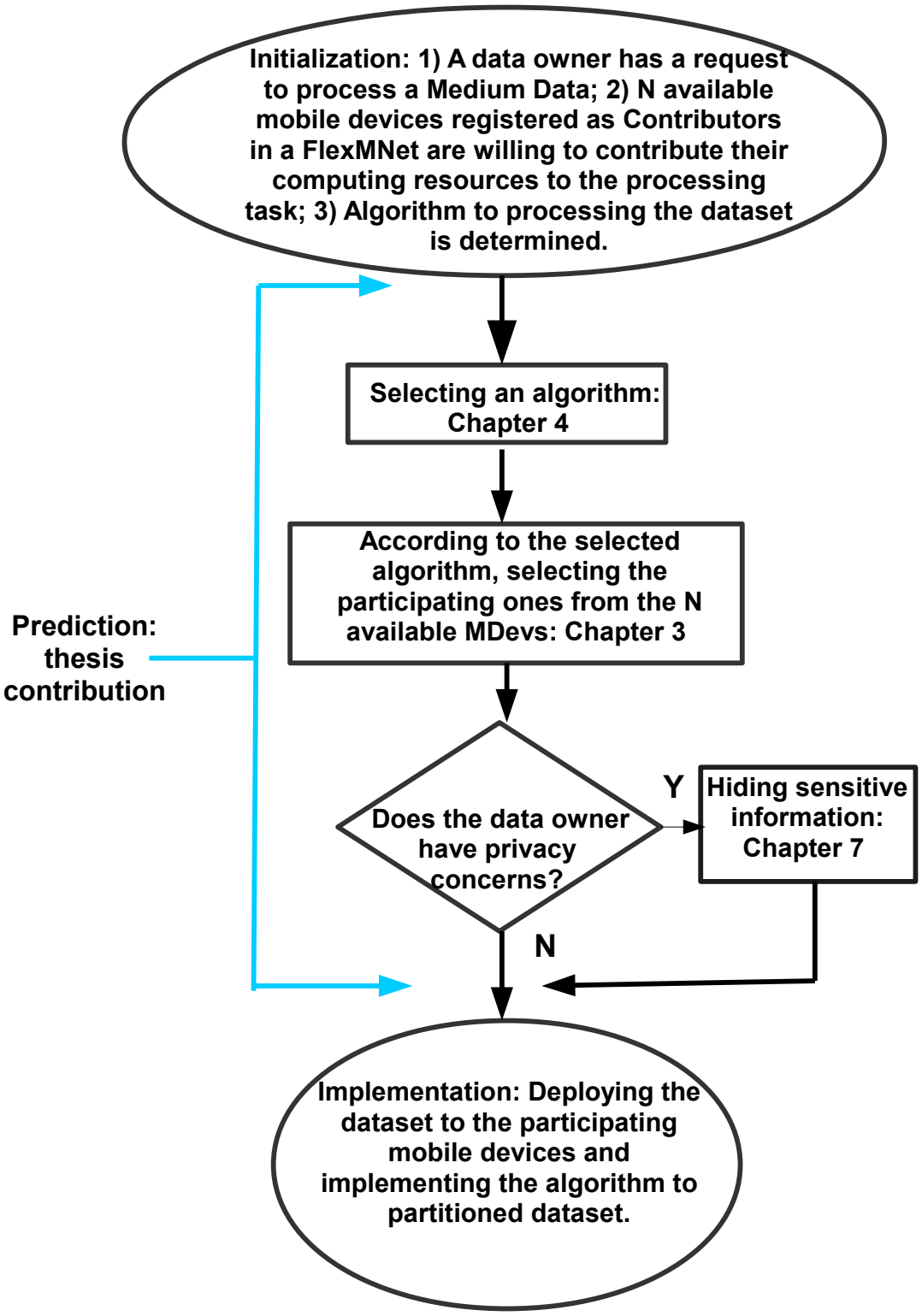

Figure 8.1. Medium data processing over FlexMNet and my contributions

Due to the slow transmission speed over FlexMNet, distributing the appropriate amount of data proportional to the computing capability of each mobile device in FlexMNet is important for improving performance. Appropriate distribution strategies benefits all devices in completing their local computing at a similar time in each iteration, which further improves the overall performance of the entire processing. This research proposes a strategy for predicting the optimised groups from which mobile devices should be selected to form a task-processing network. When distributing a data set into mobile devices, there is a probability that each mobile device is allocated with an inappropriate amount of data, which is either beyond or under its computing capability. Through predicting the optimised combination, this probability could be decreased.

FlexMNet is limited to process *Medium Data*, which has been defined in Chapter 3.

Apart from finding the most appropriate amount of data for each mobile device to process, performance of the processing could be improved from optimising (a) distributed algorithm(s) which applies to a processing task over FlexMNet. When there are a number of algorithms available for a processing task, an optimized algorithm could be determined/selected according to various criteria. Chapter 4 discusses a number of criteria and presents a ranking strategy to determine an optimized algorithm for a certain processing task. In Chapter 5, an experiment is devised in a simulated environment comprising a number of mobile devices, which demonstrates the usability of the prediction strategy, presented in Chapters 3 and 4. The experiment is conducted with the goal of predicting a group of mobile devices with the least processing time to process a given dataset.

Organization of a group of mobile devices is another factor to affect the performance over FlexMNet. Research on the organisation is presented as a framework of FlexMNet (see Chapter 6). The framework provides a platform to make application (app) or plug-in development for mobile devices possible in the future. A followed case study in medical environment presents one of the possible applications of this framework. The proposal of a framework also benefits the application of a general data processing algorithm over FlexMNet. To illustrate and discuss this framework, distributed data mining strategies are adopted as an example for processing the *medium data*.

Privacy and sensitive information are concerned by any processing network. Strategies to protect sensitive information stored in each mobile device are adopted. These strategies become another factor which may affect the performance over FlexMNet. Chapter 7 discusses possible causes of affecting the performance when protecting sensitive information in the framework. It also proposes a six-Bands of Privacy Preserving Assets (the BPPA) which used to minimize the sensitive information in order to improve the performance. These strategies are also used to assist the framework to determine the optimized strategy for a data processing task when there are more than one privacy protection strategies available.

The following sections conclude the thesis by discussing future directions in chapters.

8.1 Future Work in Chapters 3, 4 and 5

The prediction strategy proposed in Chapter 3 uses a formula to predict the processing time and memory usage during a data processing procedure within a group of mobile devices. The aim of proposing the prediction formula is to simplify the calculation due to limited computing resources in mobile devices. This formula adopts a number of basic math operations, such as Plus and Minus ($+$, $-$), Division and Multiplication ($\div$, $\times$). However, the accuracy of the prediction is also affected. Currently proposed performance prediction strategies for distributed wireless computing adopted many complicated operations, such as regression models (Lee & Schopf 2003) and Matrix (Seong 2014). These complicated operations require a heavy computing load. These prediction strategies provide reasonable/tolerable results for accuracy but not for performance, especially within a group of mobile devices with limited computing resources. Consequently, they are not suitable for FlexMNet. Future work could concentrate on improving the accuracy of the formula by modifying the existing strategies under the constraint of minimizing calculations for each mobile device.

An experiment has been implemented over a router-based Wi-Fi network. The experiment demonstrated the usage of the prediction strategy. Apart from router-based networks, various other wireless networks with different settings could also be used to

undertake processing tasks over FlexMNet. A hotspot-based Wi-Fi network launched by a mobile device should be specially targeted as the experimental network. The reason is that a self-launched hotspot by a mobile device is becoming one of the common ways to form a Wi-Fi network in the situation that data owners require results urgently. Experimental results collected from different networks could be used to improve both the accuracy and performance of the prediction strategy. Additionally, further experiments could work on more number of mobile devices.

Experiment in this research only consider a similar distance from each mobile device to the data owner with a high level stability of the network. In real world scenarios, distances from mobile devices to a data owner are often different. Further experiments should consider the differences of distance and latency in networks with low-level stability.

Experiment adopts the distributed version of a data mining algorithm, k-means, which exchanges intermediate results twice between mobile devices. Future experiments could adopt algorithms that exchanging results between devices as many times as the results show convergence. The data set used in the experiment has been horizontally partitioned to each mobile device. However, when a data set is vertically partitioned, both the frequency of exchanging subresults between mobile devices and the complexity of these intermediate results are increased. The effectiveness of the prediction strategy in this situation needs to be further studied.

8.2 Future Work in Chapter 6

The medium data processing framework of FlexMNet comprises different *Contributors*. As the FlexMNet is an ad hoc network and the participation of each mobile device is based on their own decisions, self configuration of the network and intelligence of Contributors are the future research directions of the framework. Contributors could have more intelligent behaviors to interact with other contributors in order to balance the workload during the procedure of processing tasks. For example, how a Resource Contributor (RC) determines the most proper Algorithm Contributor to accommodate

the RC into a specific scenario and how a Facilitator Contributor determines the data distribution strategy with the best overall performance for different scenarios.

There are still lots of details need to be considered and discussed to implement this application, such as cooperation with various mobile devices, integration or communication of different local networks and communication and security protocols. Implementation of this framework is beyond the scope of this thesis, which however could be the future work of this research.

In summary, further study of FlexMNet focuses on the intelligence of the framework, which further improves performance of computing-intensive tasks.

8.3 Future Work in Chapter 7

The BPPA provides an opportunity for the framework of FlexMNet to minimising sensitive information for data owners. However, the BPPA has not been applied in real scenarios, including getting responses regarding the efficiency and usability of the BPPA from data owners and developing apps or applicable tools for the BPPA.

Performance improvement of computing-intensive tasks over FlexMNet is the focus of this thesis, not privacy protection. The BPPA is an optional technology for the framework in order to determine a minimum amount of sensitive information for a computing task. The BPPA could be substituted by other technologies to protect sensitive information which could provide better outcome to adapt in an instantiated system by the framework. However, sensitive information over FlexMNet could be studied as an independent research direction because of the importance of privacy protection in any data processing system.

This chapter also proposes a number of criteria and prediction strategy with a number of factors to be considered. These criteria and the prediction strategy are used to assist data owners to select the most suitable privacy protection algorithm to apply on their data set. Future work could be done on refining the proposed prediction strategies and associated criteria in different scenarios. Furthermore, these criteria and formula are designed in terms of the issues presented or discussed in a number of publications.

However, they have not been tested in real scenarios. The refinement and improvement could be made in terms of feedback from customers experience in the real world.

Bibliography

Aggarwal, C. C. & Yu, P. S., eds (2008), *Privacy-preserving data mining: models and algorithms*, Springer Science+Business Media, LLC.

Agrawal, R., Imieliński, T. & Swami, A. (1993), 'Mining association rules between sets of items in large databases', *ACM SIGMOD Record* **22**(2), 207–216.

Agrawal, R. & Shafer, J. (1996), 'Parallel mining of association rules', *IEEE Transactions on Knowledge and Data Engineering* **8**, 962–969.

Agrawal, R. & Srikant, R. (1994), Fast algorithms for mining association rules, *in* 'the 20th International Conference Very Large Data Bases', Vol. 1215 of *VLDB 94*, Morgan Kaufmann Publishers, Santiago, Chile, pp. 487–499.

Agrawal, V. (2015), Security and privacy issues in wireless sensor networks for healthcare, *in* R. Giaffreda, R.-L. Vieriu, E. Pasher, G. Bendersky, A. J. Jara, J. J. Rodrigues, E. Dekel & B. Mandler, eds, 'Internet of Things. User-Centric IoT. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering', Vol. 150, Springer International Publishing, Cham, pp. 223–228.

Albashiri, K. A. (2013), 'Data partitioning and association rule mining using a multi-agent system using a multi-agent system', *International Journal of Engineering Science and Innovative Technology (IJESIT)* **2**(5), 161–169.

Allen, G. W., Johnson, J., Ruiz, M., Lees, J. & Welsh, M. (2005), Monitoring volcanic eruptions with a wireless sensor network, *in* 'the 2nd European Workshop on Wireless Sensor Networks', EWSN 2005, IEEE, Istanbul, Turkey, pp. 108–120.

Anjum, A., Aizad, S., Arshad, B., Subhani, M., Davies-Tagg, D., Abdullah, T. & Antonopoulos, N. (2017), *Big Data Analytics in Healthcare: A Cloud-Based Framework for Generating Insights*, Springer International Publishing, Cham, pp. 153–170.

Ariwa, E., Senousy, M. & Gaber, M. M. (2003), 'Facilities management and e-business model application for distributed data mining using mobile agents', *The International Journal of Applied Marketing* **2**(1), 123–145.

Arlia, D. & Coppola, M. (2001), Experiments in parallel clustering with DBSCAN, *in* 'the 7th International Euro-Par Conference Manchester on Parallel Processing', Euro-Par '01, Springer-Verlag, Manchester, UK, pp. 326–331.

Ashrafi, M. Z., David, T. & Smith, K. (2004), 'ODAM: an optimized distributed association rule mining algorithm', *IEEE Distributed Systems Online* **5**(3), 1–18.

Attiya, H. & Welch, J. (2004), *Distributed Computing-Fundamentals, Simulations and Advanced Topics*, 2nd edn, John Wiley & Sons, Inc. Hoboken, New Jersey.

Baldassari, J., Finkel, D. & Toth, D. (2006), SLINC: A framework for volunteer computing, *in* 'the 18th IASTED International Conference on Parallel and Distributed Computing and Systems', PDCS 2006, ACTA Press, Dallas, TX, pp. 540–545.

Bandyopadhyay, S., Giannella, C., Maulik, U., Kargupta, H., Liu, K. & Datta, S. (2006), 'Clustering distributed data streams in peer-to-peer environments', *Information Sciences* **176**, 1952 – 1985.

Basagni, S., Conti, M., Giordano, S. & Stojmenovic, I., eds (2004), *Mobile Ad Hoc Networking*, The Institute of Electrical and Electronics Engineers, Inc.

Belbachir, K. & Belbachir, H. (2012), The parallelization of algorithm based on partition principle for association rules discovery, *in* 'International Conference on Multimedia Computing and Systems', ICMCS 2012, IEEE, Tangiers, Morocco, pp. 1–6.

Bhamra, G. S., Verma, A. K. & Patel, R. B. (2015), 'Agent based frameworks for distributed association rule mining: an analysis', *International Journal in Foundations of Computer Science & Technology (IJFCST)* **5**(1).

Branch, J., Szymanski, B., Wolff, R., Gianella, C. & Kargupta, H. (2006), In-network outlier detection in wireless sensor networks, *in* 'the 26th International Conference on Distributed Computing Systems', ICDCS'06, IEEE, Lisboa, Portugal, pp. 102–111.

Buyya, R. (1999), *High Performance Cluster Computing: Architecutres and Systems*, Prentice Hall PTR.

Buyya, R., Yeo, C. & Venugopal, S. (2008), Market-oriented cloud computing: vision, hype, and reality for delivering it services as computing utilities, *in* 'the 10th IEEE International Conference on High Performance Computing and Communications', IEEE, Dalian, China, pp. 5–13.

Calandrino, J. A., Kilzer, A., Narayanan, A., Felten, E. W. & Shmatikov, V. (2011), "You might also like:" privacy risks of collaborative filtering, *in* 'IEEE Symposium on Security and Privacy (SP)', IEEE, The Claremont Resort, Oakland, California, USA, pp. 231–246.

Cannataro, M., Congiusta, A., Pugliese, A., Talia, D. & Trunfio, P. (2004), 'Distributed data mining on grids: services, tools, and applications', *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **34**(6), 2451–2465.

Chan, H. & Perrig, A. (2003), 'Security and privacy in sensor networks', *Computer* **36**(10), 103–105.

Chen, H., Chiang, R. & Storey, V. (2012), 'Business intelligence and analytics: From big data to big impact', *MIS Quarterly: Management Information Systems* **36**(4), 1165–1188.

Chess, D. M., Harrison, C. G. & Kershenbaum, A. (1996), Mobile agents: Are they a good idea?, *in* 'Mobile Object Systems'96', pp. 25–45.

Cheung, A. S. & Weber, R. H., eds (2015), *Privacy and legal issues in cloud computing*, Edward Elgar Publishing Limited, Cheltenham,UK.

Chun, B.-G. & Maniatis, P. (2009), Augmented smartphone applications through clone cloud execution, *in* 'the 12th Conference on Hot Topics in Operating Systems', HotOS'09, USENIX Association, Berkeley, CA, USA, pp. 8–13.

Cisco (2017), *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 20162021 White Paper*.
URL: *cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html*

Codd, E. F. (1970), 'A relational model of data for large shared data banks', *Communication ACM* **13**(6), 377–387.

Comito, C., Falcone, D., Talia, D. & Trunfio, P. (2011), Energy efficient task allocation over mobile networks, *in* '2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing', pp. 380–387.

Comito, C., Falcone, D., Talia, D. & Trunfio, P. (2013*a*), Intelligent distributed computing vi: Studies in computational intelligence, Springer-Verlag, Berlin, Heidelberg, chapter A Distributed Allocation Strategy for Data Mining Tasks in Mobile Environments, pp. 231–240.

Comito, C., Falcone, D., Talia, D. & Trunfio, P. (2013*b*), 'Scheduling data mining applications in mobile computing environments', *ERCIM News* pp. 15–16.

Comito, C., Falcone, D., Talia, D. & Trunfio, P. (2017), 'Energy-aware task allocation for small devices in wireless networks', *Concurrency and Computation: Practice and Experience* **29**(1), 3831–3855.

Comito, C. & Talia, D. (2014), Energy-aware clustering of ubiquitous devices for collaborative mobile applications, *in* 'the 6th International Conference on Mobile Computing, Applications and Services', pp. 133–142.

Congiusta, A., Talia, D. & Trunfio, P. (2005), *Data Mining and Knowledge Discovery Handbook*, Springer International Publishing, chapter 48 Parallel and Grid-Based Data Mining, pp. 1017–1041.

Congiusta, A., Talia, D. & Trunfio, P. (2008), 'Service-oriented middleware for distributed data mining on the grid', *Journal of Parallel and Distributed Computing* (1), 3–15.

Coulouris, G., Dollimore, J., Kindberg, T. & Blair, G. (2011), *Distributed Systems: Concepts and Design*, 5th edn, Boston: Addison-Wesley.

Datta, S., Bhaduri, K., Giannella, C., Wolff, R. & Kargupta, H. (2006), 'Distributed data mining in peer-to-peer networks', *IEEE Internet Computing* **10**, 18–26.

Diebold, F. X. (2012), 'A personal perspective on the origin(s) and development of 'big data': The phenomenon, the term, and the discipline, second version', *SSRN Electronic Journal* .

Dillon, T., Wu, C. & Chang, E. (2010), Cloud computing: Issues and challenges, *in* '24th IEEE International Conference on Advanced Information Networking and Applications', IEEE, Perth, Australia, pp. 27–33.

Ding, J. (2010), *Advances in Network Management*, CRC Press, Taylor & Francis Group, chapter 2: Evolution of Networks, pp. 3–42.

Dinh, H. T., Lee, C., Niyato, D. & Wang, P. (2013), 'A survey of mobile cloud computing: architecture, applications, and approaches', *Wireless Communications and Mobile Computing* **13**(18), 1587–1611.

Duarte, M. F. & Hu, Y. H. (2004), 'Vehicle classification in distributed sensor networks', *Journal of Parallel and Distributed Computing* **64**, 826–838.

Evfimievski, A., Gehrke, J. & Srikant, R. (2003), Limiting privacy breaches in privacy preserving data mining, *in* 'the 22nd SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems', PODS '03, ACM, San Diego, California, pp. 211–222.

Fayyad, U. M., Piatetsky-Shapiro, G. & Smyth, P. (1996*a*), From data mining to knowledge discovery: an overview, *in* U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth & R. Uthurusamy, eds, 'Advances in knowledge discovery and data mining', American Association for Artificial Intelligence, Menlo Park, CA, USA, pp. 1–34.

Fayyad, U., Piatetsky-shapiro, G. & Smyth, P. (1996*b*), 'From data mining to knowledge discovery in databases', *AI Magazine* **17**, 37–54.

Fordham University, D. o. C. & Science, I. (2012), 'WISDM (Wireless Sensor Data Mining) project'.
URL: *storm.cis.fordham.edu/ gweiss/ wisdm/*

Forman, G. & Zhang, B. (2000), 'Distributed data clustering can be efficient and exact', *SIGKDD Explorer Newsletter* **2**(2), 34–38.

Fortino, G. & Trunfio, P. (2014), *Internet of Things Based on Smart Objects:Technology, Middleware and Applications*, Springer International Publishing, Switzerland.

Foti, D., Lipari, D., Pizzuti, C. & Talia, D. (2000), Scalable parallel clustering for data mining on multicomputers, *in* 'the 3rd International Workshop on High Performance Data Mining', Vol. 1800 of *Lecture Notes of Computer Science*, Springer-Verlag Cancun, pp. 390–398.

Fu, Y. (2001), 'Distributed data mining: An overview', *Newsletter of the IEEE Technical Committee on Distributed Processing* pp. 5–9.

Gaber, M. M., Stahl, F. & Gomes, J. B. (2014), *Pocket Data Mining: Big Data on Small Devices*, Springer International Publishing, Switzerland.

Gandomi, A. & Haider, M. (2015), 'Beyond the hype: Big data concepts, methods, and analytics', *International Journal of Information Management* **35**(2), 137 – 144.

Gardner-Stephen, P. & Challans, R. (2010), *The Serval Project*, Flinders University. **URL:** *www.servalproject.org/*

Gardner-Stephen, P., Challans, R., Lakeman, J., Bettison, A., Gardner-Stephen, D. & Lloyd, M. (2013), The serval mesh: A platform for resilient communications in disaster & crisis, *in* '2013 Global Humanitarian Technology Conference (GHTC)', IEEE, San Jose, CA, USA, pp. 162–166.

Garg, P. & Sharma, V. (2013), 'Secure data storage in mobile cloud computing', *International Journal of Scientific & Engineering Research* **4**(4), 1154–1159.

Geier, J. (2001), *Wireless LANs*, 2nd edn, Sams Publishing.

Gill, P. S. (2006), *Operating System Concepts*, Firewall Media, New Delhi, India.

Goel, A. (2012), *Computer Fundamentals*, Cengage Learning, Boston, MA.

Goh, J. Y. & Taniar, D. (2005), An efficient mobile data mining model, *in* J. Cao, L. Yang, M. Guo & F. Lau, eds, 'Parallel and Distributed Processing and Ap-

plications', Vol. 3358 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 54–58.

Golle, P. (2006), Revisiting the uniqueness of simple demographics in the us population, *in* 'the 5th ACM Workshop on Privacy in Electronic Society', WPES '06, ACM, Alexandria, VA, pp. 77–80.

Gordon, A., ed. (2015), *Official (ISC) guide to the CISSP CBK*, 4th edn, CRC Press, Taylor & Francis Group.

Grand View Research (2015), Personal/consumer electronics market analysis by product (smartphones, tablets, desktops, laptops/notebooks, digital cameras, hard disk drives, e-readers) and segment forecasts to 2020, Technical report, Grand View Research.
URL: *grandviewresearch.com/industry-analysis/personal-consumer-electronics-market*

Gray, R. S., Kotz, D., Nog, S., Rus, D. & Cybenko, G. (1996), Mobile agents for mobile computing, Technical report, Dartmouth College Hanover, NH, USA.

Gu, T., Wang, L., Wu, Z., Tao, X. & Lu, J. (2011), 'A pattern mining approach to sensor-based human activity recognition', *IEEE Transactions on Knowledge and Data Engineering* **23**(9), 1359–1372.

Gummadi, R., Li, X., Govindan, R., Shahabi, C. & Hong, W. (2005), 'Energy-efficient data organization and query processing in sensor networks', *SIGBED Review* **2**(1), 7–12.

Haghighi, P., Krishnaswamy, S., Zaslavsky, A., Gaber, M., Sinha, A. & Gillick, B. (2013), 'Open mobile miner: a toolkit for building situation-aware data mining applications', *Journal of Organizational Computing and Electronic Commerce* **23**(3), 224 – 248.

Halperin, D., Heydt-Benjamin, T., Ransford, B., Clark, S., Defend, B., Morgan, W., Fu, K., Kohno, T. & Maisel, W. (2008), Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses, *in* 'the 2008 IEEE

Symposium on Security and Privacy', IEEE, The Claremont Resort Oakland, California, USA, pp. 129–142.

Harrington, J. L. (2016), *Relational Database Design and Implementation*, 4th edn, Elsevier Science.

Hashemi, S. M. & Bardsiri, A. K. (2012), 'Cloud computing vs. grid computing', *ARPN Journal of Systems and Software* **2**(5), 188–194.

Honava, V., Miller, L. & Wong, J. (1998), Distributed knowledge networks, *in* 'IEEE Information Technology Conference, Information Environment for the Future', IEEE, Syracuse, NY, USA, pp. 71–74.

Huang, D., Zhang, X., Kang, M. & Luo, J. (2010), Mobicloud: Building secure cloud framework for mobile computing and communication, *in* 'the 5th IEEE International Symposium on Service Oriented System Engineering', IEEE, Nanjing, China, pp. 27–34.

Huang, D.and Zhou, Z., Xu, L., Xing, T. & Zhong, Y. (2011), Secure data processing framework for mobile cloud computing, *in* '2011 IEEE Conference on Computer Communications Workshops', IEEE, Shanghai, China, pp. 614–618.

Huerta-Canepa, G. & Lee, D. (2010), A virtual cloud computing provider for mobile devices, *in* 'the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond', ACM, San Francisco, California, pp. 1–5.

Hull, B., Bychkovsky, V., Chen, K., Goraczko, M., Miu, A., Shih, E.and Zhang, Y., Balakrishnan, H. & Madden, S. (2006), Cartel: A distributed mobile sensor computing system, *in* 'the 4th International Conference on Embedded Networked Sensor Systems', ACM, Boulder, Colorado, USA, pp. 125–138.

Imielinski, T. & Korth, H. F., eds (1996), *Mobile Computing*, Springer US.

Indrawan, M., Krishnaswamy, S. & Ranjan, T. (2003), Using mobile agents to support unreliable database operations, *in* 'the 17th International Conference on Advanced Information Networking and Applications', AINA2003, IEEE, Xi'an, China.

Jackson, J. (2002), 'Data mining: A conceptual overview', *Communications of the Association for Information Systems* **8**, 267–296.

Jia, W. & Wanlei, Z. (2005), *Distributed Network Systems: From Concepts to Implementations*, Springer Science & Business.

Johnson, D. B. & Maltz, D. A. (1996), *Mobile Computing*, Vol. 353, Springer US, chapter 5: Dynamic Source Routing in Ad Hoc Wireless Networks, pp. 153–181.

Johnson, E. L. & Kargupta, H. (1999), Collective, hierarchical clustering from distributed, heterogeneous data, *in* M. J. Zaki & C.-T. Ho, eds, 'Revised Papers from Large-Scale Parallel Data Mining, Workshop on Large-Scale Parallel KDD Systems, SIGKDD', Vol. 1759, Springer-Verlag, London, UK, pp. 221–244.

Juang, P., Oki, H., Wang, Y., Martonosi, M., Peh, L. & Rubenstein, D. (2002), 'Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet', *SIGOPS Operating Systems Review* **36**(5), 96–107.

Kargupta, H., Park, B.-H., Pittie, S., Liu, L., Kushraj, D. & Sarkar, K. (2002), 'MobiMine: monitoring the stock market from a PDA', *SIGKDD Explorations* pp. 37–46.

Kargupta, H., Puttagunta, V., Klein, M. & Sarkar, K. (2006), 'On-board vehicle data stream monitoring using minefleet and fast resource constrained monitoring of correlation matrices', *New Generation Computing* **25**(1), 532.

Kaur, N. & Monga, S. (2014), 'Comparisons of wired and wireless networks: A review', *International Journal of Advanced Engineering Technology* **11**, 34–35.

Khan, M. A. & Sivrikaya, F. (2015), Cloud computing: Futrure of the past, *in* R. S. Segall, J. S. Cook & Q. Zhang, eds, 'Research and Applications in Global Supercomputing', IGI Global, Hershey, USA, pp. 228–302.

Kholod, I., Kuprianov, M. & Petukhov, I. (2016), Parallel and distributed data mining in cloud, *in* P. Perner, ed., 'Advances in Data Mining. Applications and Theoretical Aspects', Springer International Publishing, Cham, pp. 349–362.

Konopka, G. (1963), *Social Group Work: A Helping Process.*, Prentice Hall, Englewood Cliffs, NT.

Kriegel, H. P., Kroger, P., Pryakhin, A. & Schubert, M. (2005), Effective and efficient distributed model-based clustering, *in* 'the 5th IEEE International Conference on Data Mining (ICDM'05)', IEEE, Houston, TX, USA, pp. 258–265.

Kufrin, R. (1997), Generating c4.5 production rules in parallel, *in* 'Proceedings of the 14th National Conference on Artificial Intelligence', AAAI Press, Rhode Island, pp. 565–570.

Kung, H. T. & Vlah, D. (2003), 'Efficient location tracking using sensor networks', *IEEE Transaction on Wireless Communication and Networking* **3**, 1954–1961.

Kurose, J. F. & Ross, K. W. (2013), *Computer Networking A Top-Down Approach*, 6th edn, PEARSON Education, Inc., Boston, MA.

Lack, L., Scott, H., Micic, G. & Lovato, N. (2017), 'Intensive sleep re-training: From bench to bedside', *Brain Sciences* **7**(4).

Laney, D. (2001), '3-d data management: Controlling data volume, velocity and variety'.
URL: *blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf*

Le-Khac, N.-A., Aouad, L. & Kechadi, M. M.-T. (2007), A new approach for distributed density based clustering on grid platform, *in* R. Cooper & J. Kennedy, eds, 'Data Management. Data, Data Everywhere. BNCOD 2007', Vol. 4587 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg.

Lee, B.-D. & Schopf, J. M. (2003), Run-time prediction of parallel applications on shared environments, *in* '2003 Proceedings IEEE International Conference on Cluster Computing', pp. 487–491.

Lee, J., Choi, S., Lim, J., Suh, T., Gil, J. & Yu, H. (2010), Mobile grid system based on mobile agent, *in* T.-h. Kim, S. S. Yau, O. Gervasi, B.-H. Kang, A. Stoica & D. Ślezak, eds, 'Grid and Distributed Computing, Control and Automation. Communications in Computer and Information Science.', Vol. 121, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 117–126.

Li, R., de Vries, D. & Roddick, J. (2011), Bands of privacy preserving objectives: Classification of ppdm strategies, *in* 'the 9th Australasian Data Mining Conference - Volume 121', AusDM '11, Australian Computer Society, Inc., Darlinghurst, Australia, pp. 137–152.

Li, X. F., Wang, J. H. & Gao, W. W. (2013), Examination system in the cloud computing platform based on data mining, *in* 'International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC)', pp. 1605–1608.

Lokeswari, Y. V. & Jacob, S. G. (2016), A comparative study on parallel data mining algorithms using hadoop map reduce: A survey, *in* 'the 2nd International Conference on Information and Communication Technology for Competitive Strategies', ICTCS '16, ACM, Udaipur, Rajasthan, India, pp. 143:1–143:6.

MainWaring, A., Polastre, J., Szewczyck, R., Culler, D. & Anderson, J. (2002), Wireless sensor networks for habitat monitoring, *in* 'the 1st ACM International Workshop on Wireless Sensor Networks and Applications', WSNA '02, ACM, Atlanta, GA, USA, pp. 88–97.

Mateo, R. M., Cervantes, L. F., Yang, H.-K. & Lee, J. (2007), Mobile agents using data mining for diagnosis support in ubiquitous healthcare, *in* 'the 1st KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications', KES-AMSTA 07, Springer-Verlag, Berlin, Heidelberg, Wroclaw, Poland, pp. 795–804.

McConnell, J. J. (2008), *Analysis of Algorithms*, Jones & Bartlett Learning.

Menon, S., Sarkar, S. & Mukherjee, S. (2005), 'Maximizing accuracy of shared databases when concealing sensitive patterns', *Information System Research* **16**, 256–270.

Moemeng, C., Gorodetsky, V., Zuo, Z., Yang, Y. & Zhang, C. (2009), Agent-based distributed data mining: A survey, *in* L. Cao, ed., 'Data Mining and Multi-agent Integration', Springer US, pp. 47–58.

Nambi, S. N. A. U., Vasirani, M., Prasad, R. V. & Aberer, K. (2014), Performance analysis of data processing architectures for the smart grid, *in* 'IEEE PES Innovative

Smart Grid Technologies, Europe', IEEE, Istanbul, Turkey, pp. 1–6.

Nestorov, S. & Jukic, N. (2003), Ad hoc association-rule mining within the data warehouse, *in* 'the 36th Hwaii International Conference on System Sciences', HICSS'03, IEEE, Big Island, HI, USA, pp. 232.1–10.

O. Kwon, N. Lee, B. S. (2014), 'Data quality management, data usage experience and acquisition intention of big data analytics', **34**(3), pp. 387–394.

Oliveira, S. R. M. & Zaïane, O. R. (2003), Algorithms for balancing privacy and knowledge discovery in association rule mining, *in* 'the 7th International Database Engineering and Applications Symposium', IEEE, Hong Kong, pp. 54–63.

Otey, M. E., Wang, C., Parthasarathy, S., Veloso, A. & Meira, W. (2003), Mining frequent itemsets in distributed and dynamic databases, *in* 'the 3rd IEEE International Conference on Data Mining', IEEE, Melbourne, Florida, USA, pp. 617–620.

Park, B. & Kargupta, H. (2003), Distributed data mining: Algorithms, systems, and applications, *in* N. Ye, ed., 'The Handbook of Data Mining', Lawrence Erlbaum Associates., pp. 341–358.

Park, H., Basaran, C., Park, T. & Son, S. H. (2014), 'Energy-efficient privacy protection for smart home environments using behavioral semantics', *Sensors, Online published by MDPI* **14**(9).

Parmar, K. B., Jani, N. N., Shrivastav, P. S. & Patel, M. H. (2013), 'Mobile grid computing: Facts or fantasy?', *International Journal of Multidisciplinary Sciences and Engineering* **4**(1).

Patient Empowerment Network (2015), 'Can digital wearables help in clinical trials?'. **URL:** *powerfulpatients.org/2015/10/06/can-digital-wearables-help-in-clinical-trials/*

Perkins, C. & Bhagwat, P. (1994), 'Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers', *ACM SIGCOMM Computer Communication Review* **24**(4), 234–244.

Phan, T., Huang, L. & Dulan, C. (2002), Challenge: Integrating mobile wireless devices into the computational grid, *in* 'the 8th Annual International Conference on Mobile Computing and Networking', MobiCom '02, ACM, Atlanta, Georgia, USA, pp. 271–278.

Pittie, S., Kargupta, H. & Park, B. (2003), 'Dependency detection in mobimine: a systems perspective', *Information Sciences* **155**(3-4), 227–243.

Piwek, L., Ellis, D. A., Andrews, S. & Joinson, A. (2016), 'The rise of consumer health wearables: Promises and barriers', *PLoS Medicine* **13**(2).

Pluntke, C., Eggert, L. & Kiukkonen, N. (2011), Saving mobile device energy with multipath TCP, *in* 'the 3rd ACM international Workshop on MobiArch', ACM, Bethesda, MD, USA, pp. 1–6.

Poo, D., Kiong, D. & Ashok, S. (2008), *Object-Oriented Programming and Java*, 2nd edn, Springer-Verlag London.

Prabhu, C. (2008), *Grid and Cluster Computing*, PHI Learning Private Limited, Delhi.

Price, R. (2007), *Fundamentals of Wireless Networking*, The McGraw-Hill Companies, Inc.

Price Waterhouse Coopers (2011), 'Protect your organization's sensitive information and reputation with high-risk data discovery'.
URL: *pwc.com/us/en/it-risk-security/assets/high-risk-data-discovery.pdf*

Rafrastara, F. A. & Deyu, Q. (2016), 'A survey and taxonomy of distributed data mining research studies: a systematic literature review', *International Journal of Computer Science and Information Security (IJCSIS)* **14**(5), 12–30.

Rahman, M., Gao, J. & Tsai, W. T. (2013), Energy saving in mobile cloud computing, *in* '2013 IEEE International Conference on Cloud Engineering (IC2E)', IEEE, Redwood City, CA, USA, pp. 285–291.

Savasere, A., Omiecinski, E. & Navathe, S. (1995), An efficient algorithm for mining association rules in large databases, *in* 'the 21th International Conference on Very Large Data Bases', VLDB '95, Morgan Kaufmann Publishers Inc., Zurich, Switzerland, pp. 432–444.

Schroeck, M., Shockley, R., Smart, J., Romero-Morales, D. & Tufano, P. (2012), 'Analytics: The real-world use of big data. how innovative enterprises extract value from uncertain data'. Executive Report.
URL: *www-935.ibm.com/services/multimedia/*

Senn, J. (2000), 'The emergence of m-commerce', *IEEE Computer* **33**, 148–150.

Seong, Jin-Taekand Lee, H.-N. (2014), 'Predicting the performance of cooperative wireless networking schemes with random network coding', *IEEE Transactions on Communication* **62**(8).

Shinde, S. (2009), *Computer Network*, New Age International (P) Ltd., Publishers.

Silva, J. C. d., Giannella, C., Bhargava, R., Kargupta, H. & Klusch, M. (2005), 'Distributed data mining and agents', *Engineering Applications of Artificial Intelligence* **18**(7), 791–807.

Sinha, K., Ghosh, S. C. & Sinha, B. P. (2016), *Wireless Networks and Mobile Computing*, CRC Press, Taylor & Francis Group.

Sipser, M. (2010), *Introduction to the Theory of Computation*, 3rd edn, Dorling Kindersley (India) Pvt. Ltd., New Delhi, India.

Srividya, M. & Vijayarani, N. (2015), 'Wired vs wireless using advanced network', *International Journal of Engineering Research and General Science* **3**(2), 825–832.

Stahl, F., Gaber, M. M., Aldridge, P., May, D., Liu, H., Bramer, M. & Yu, P. S. (2012), Transactions on large-scale data- and knowledge-centered systems v, Springer-Verlag, Berlin, Heidelberg, chapter Homogeneous and Heterogeneous Distributed Classification for Pocket Data Mining, pp. 183–205.

Sweeney, L. (2000), Uniqueness of simple demographics in the u.s. population, Technical report, Carnegie Mellon University, Laboratory for International Data Privacy, Pittsburgh, PA.

Talia, D. & Trunfio, P. (2008), *Service-Oriented Architectures for Distributed and Mobile Knowledge Discovery*, CRC Press, pp. 223–242.

Talia, D. & Trunfio, P. (2010), Mobile data mining on small devices through web services, *in* L. Yang, A. Waluyo, J. Ma, L. Tan & B. Srinivasan, eds, 'Mobile Intelligence', John Wiley & Sons, Inc., pp. 264–276.

Tan, P.-N., Steinbach, M. & Kumar, V. (2006), *Introduction to Data Mining*, Addison-Wesley, Boston, MA.

Tanenbaum, A. & Wetherall, D. J. (2011), *Computer Networks*, 5th edn, Prentice Hall.

The BOINC project (2017).
URL: *//boinc.berkeley.edu/*

Toh, C. K. (2001), *Ad Hoc Mobile Wireless Networks: Protocols and Systems*, 1st edn, Prentice Hall PTR, Upper Saddle River, NJ, USA.

Trunfio, P., Talia, D., Papadakis, H., Fragopoulou, P., Mordacchini, M., Pennanen, M., Popov, K., Vlassov, V. & Haridi, S. (2007), 'Peer-to-peer resource discovery in grids: Models and systems', *Future Generation Computer Systems* **23**(7), 864–878.

Tsoumakas, G. & Vlahavas, L. (2008), Distributed data mining, *in* J. Wang, ed., 'Encyclopedia of Data Warehousing and Mining (2nd Edition)', Idea Group Reference, pp. 709–715.

Umar, A. (2004), *Mobile Computing And Wireless Communications: Applications, Networks, Platforms, Architectures and Security*, NGE Solutions, Inc., US.

Vallina-Rodriguez, N. & Crowcroft, J. (2011), ErdOs: achieving energy savings in mobile devices, *in* 'the 6th International Workshop on MobiArch', MobileArch'11, ACM, Bethesda, Maryland, USA, pp. 37–42.

Verykios, V. S., Bertino, E., Fovino, I. N., Provenza, L. P., Saygin, Y. & Theodoridis, Y. (2004), 'State-of-the-art in privacy preserving data mining', *SIGMOD Record* **33**, 50–57.

Vivaldi, F. (2001), *Experimental Mathematics with Maple*, Chapman & Hall/CRC Press, Boca Raton, Florida.

Wang, F. & Helian, N. (2005), Mining global association rules on an oracle grid by scanning once distributed databases, *in* J. Cunha & P. Medeiros, eds, 'European

Conference on Parallel Processing', Vol. 3648 of *Lecture Notes in Computer Science*, Springer, Berlin, Heidelberg, pp. 370–378.

Wang, F., Helian, N., Guo, Y. & Jin, H. (2003), A distributed and mobile data mining system, *in* 'the 4th International Conference on Parallel and Distributed Computing, Applications and Technologies', PDCAT'2003, IEEE, Chengdu, China, pp. 916–918.

Wang, Y., Chen, I.-R. & Wang, D.-C. (2015), 'A survey of mobile cloud computing applications: Perspectives and challenges', *Wireless Personal Communications: An International Journal* **80**(4), 1607–1623.

Warneke, B., Last, M., Lifebowitz, B. & Pister, K. (2001), 'Smart dust: Communicating with a cubic millimeter computer', *IEEE Computer* **34**, 4451.

Weiss, A. (2007), 'Computing in clouds', *ACM Networker* **11**(4), 18–25.

Weiss, N. A. (2016), *Introductory Statistics*, 10th edn, Pearson Education.

West, G. P. (1967), The best approach to a large computing capability, *in* 'the April 18-20, Spring Joint Computer Conference', AFIPS '67 (Spring), ACM, Atlantic City, New Jersey, pp. 467–469.

Whitman, M. E. & Mattord, H. J. (2016), *Principles of Information Security*, 5th edn, Cengage Learning, Boston, MA.

WNDW.net (2013), *Wireless networking in the developing world-A practical guide to planning and building low-cost telecommunications infrastructure*, 3rd edn.

Wu, C.-H. & Irwin, J. D. (2013), *Introduction to Computer Networks and Cybersecurity*, CRC Press, Taylor & Francis Group.

Wu, Q., Rao, N. S. V., Barhen, J., Iyengar, S. S., Vaishnavi, V. K., Qi, H. & Chakrabarty, K. (2004), 'On computing mobile agent routes for data fusion in distributed sensor networks', *IEEE Transactions on Knowledge and Data Engineering* **16**, 740–753.

Wysocki, T. A., Dadej, A. & Wysocki, B. J., eds (2005), *Advances Wired and Wireless Networks*, Springer Science & Business Media.

Yang, J., Honavar, V., Miller, L. & J.Wong (1998), Intelligent mobile agents for information retrieval and knowledge discovery from distributed data and knowledge sources, *in* 'IEEE Information Technology Conference', IEEE, Syracuse, NY, USA, pp. 99 –102.

Yang, W. & Huang, S. (2007), Privacy preserving clustering for multi-party, *in* 'the12th International Conference on Database Systems for Advanced Applications', DASFAA'07, Springer-Verlag, Bangkok, Thailand, pp. 213–224.

Yao, L. (2013), ADAGE: a framework for supporting user-driven ad-hoc data analysis processes, PhD thesis, Computer Science & Engineering, Faculty of Engineering, University of New South Walse.

Zaki, M. J. (2000), Parallel and distributed data mining: An introduction, *in* M. J. Zaki & C.-T. Ho, eds, 'Large-Scale Parallel Data Mining', Vol. 1759 of *Lecture Notes of Artificial Intelligence*, Springer-Verlag Berlin Heidelberg, pp. 1–23.

Zaki, M. J. & Ho, C.-T., eds (2000), *Large-Scale Parallel Data Mining*, Vol. 1759 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag Berlin Heidelberg.

Zaki, M. J. & Pan, Y. (2002), 'Introduction: Recent developments in parallel and distributed data mining', *Distributed and Parallel Databases* **11**(2), 123–127.

Zeng, L., Li, L., Duan, L., Lu, K., Shi, Z., Wang, M., Wu, W. & Luo, P. (2012), 'Distributed data mining: a survey', *Information Technology and Management* **13**(4), 403–409.

Zhang, C., Zhang, Z. & Cao, L. (2005), Agents and data mining: Mutual enhancement by integration, *in* V. Gorodetsky, J. Liu & V. A. Skormin, eds, 'Autonomous Intelligent Systems: Agents and Data Mining: International Workshop, AIS-ADM 2005', Springer Berlin Heidelberg, St. Petersburg, Russia, pp. 50–61.

Zhu, J. (2017), 'Research on data mining of electric power system based on hadoop cloud computing platform', *International Journal of Computers and Applications* pp. 1–7.

Zubairi, J. A., ed. (2009), *Applications of Modern High Performance Networks: Wireless Sensor Networks, Ultra Wide Band, Mobile Ad-hoc Networks, Computer*

Clusters, Smart Agriculture and Distance Computing, Bentham Science Publishers.