Scalable Computing:
Practice and Experience

Scientific International Journal
for Parallel and Distributed Computing

ISSN: 1895-1767

Universitatea de Vest din Timişoara

Volume 18(3) September 2017

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

Scalable Computing: Practice and Experience

Volume 18, Number 3, September 2017

TABLE OF CONTENTS

DOI 10.12694/scpe.v18i3.1299
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 18, Number 3, pp. iii–iv. http://www.scpe.org

INTRODUCTION TO THE SPECIAL ISSUE ON SECURE SOLUTIONS FOR NETWORK IN SCALABLE COMPUTING

This special issue aimed at incarceration of new insights, dimensions, visionaries and accomplishments achievable for security. Security is a foremost concern in scalable computing. With the progression in ICT (Information and Communication Technologies), secure solutions are the need; as the number and kinds of attacks too are progressive. Keeping this in mind, the papers selected for this issue talks about all our visualizations from the same. In todays Internet-connected world, the advance cyber-attacks are being launched on the critical infrastructure. It has shifted the pursuit of financial profit and political gains, which lead to cyber warfare on various scales. The first paper points that malware is one of the most alarming security threats being faced by the Internet today. They have the capability to circumvent the earlier developed methods of detection and mitigation which clearly shows the need of shifting from traditional cyber security to cyber threat intelligence. Authors have proposed the design of a framework for generating malware threat intelligence which has the capability to detect, analyze, and predict the malware threats and can act as an Early Warning System (EWS). The second paper talks about the current status of sentiment analysis and opinion mining focusing on the problem of sarcasm identification and detection. The article discusses the present scenario and the problem faced by the community due to the usage of sarcasm. Authors have tried a technique based on the deep convolution neural networks where they are using a single layer convolution before the classification task and claim to have higher accuracy in the classification of sarcasm as compared to existing methods.

The third paper talks about Wireless sensor networks (WSNs) that are widely used in various fields such as health monitoring, medical line, intrusion detection, and are often placed in open environment, thus vulnerable to different attacks. Various techniques were introduced to deal with security related issues of WSNs; among them trust management has been proved as an effective measure. A new protocol called Energy Efficient and Trust Aware framework for secure routing in LEACH (EETA-LEACH) has been proposed. A trust management system for WSNs has been presented to monitor the sensor nodes behaviors and evaluate their trust values based upon remaining energy, packet delivery ratio and distance. This approach is a combination of trust-based routing module and trust management module that works together to select trusted Cluster Head (CH). Simulation results have proved that proposed algorithm consumes less energy and improves packet delivery ratio.

Almost all the important services are available in the application market of Android. Unfortunately, at the same time, the prosperity of these applications also attracts abusers and malicious attackers to perform different types of attacks. The fourth paper is an exploration and all-inclusive study about various approaches to perform Android applications analysis. This gives improved identification of the problem, accessible elucidation space and possible research scope to evaluate Android devices against the possible attacks.

The fifth paper focused on the dynamics of worm propagation in the Wireless Sensor Networks (WSNs). Authors proposed a modified Susceptible-Infectious-Quarantined-Recovered-Susceptible (SIQRS) model based on epidemic theory and demonstrated the effect of quarantined state on worm propagation. This paper also describes the Stability of the worm free equilibrium and Endemic equilibrium, as well as, studies the effect of communication radius and node density.

Kavita Sharma, NIT, Kurukshetra, India
Suman Bala, Orange Labs, Meylan, France
Himani Bansal, JIIT, Noida, India
Gulshan Shrivastava, NITP, India

DOI 10.12694/scpe.v18i3.1300
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 18, Number 3, pp. 195–205. http://www.scpe.org

A FRAMEWORK FOR GENERATING MALWARE THREAT INTELLIGENCE

EKTA GANDOTRA*, DIVYA BANSAL† AND SANJEEV SOFAT‡

Abstract. Ubiquitous computing devices with network capabilities have become the critical cyber infrastructure for academia, industry and government in day-to-day life. The cyber-attacks being launched on this critical infrastructure have shifted to the pursuit of financial profit and political gains which lead to cyber warfare on various scales. The evolution of new practices like social networking, explosion of mobile devices and cloud computing have given opportunities to attackers for discovering vulnerabilities and exploiting these for creating sophisticated attacks. Malware is one of the most dreadful security threats fronting the Internet today. It is evolving and making use of new ways to target computers and mobile devices. Moreover, the exponential escalation in their volume and complexity has increased the damage caused by them. These have the capability to circumvent the earlier developed methods of detection and mitigation which clearly shows the need of shifting from traditional cyber security to cyber security intelligence. This paper purposes a design of a framework for generating Malware Threat Intelligence that can analyze, identify and predict the malware threats and can act as an Early Warning System (EWS). It also presents the real-time testing of the proposed framework which is realized by designing a prototype for providing security-as-a-service.

Key words: malware, threat intelligence, malware analysis, malware detection, malware threat level

AMS subject classifications. 68M14, 68T27

1. Introduction.

Cyberspace comprises of people, services and software linked either directly or indirectly to the Computer Networks, Internet and Telecommunications. The services and products residing on cyberspace have been adopted in almost all the sectors. Moreover, people have become habitual of services being provided by the Internet. The preservation of Confidentiality, Integrity and Availability of information and protection of critical infrastructure is the essence of secure cyberspace. During the past several years, the frequency and complexity of cyber-attacks have been changed. The evolution of new practices like social networking, explosion of mobile devices and cloud computing have given opportunities to attackers for discovering vulnerabilities and exploiting these for creating sophisticated attacks. Moreover, the new generation cyber-attacks have become more targeted, persistent and unknown. Most of these attacks are launched by people who use the Internet with wicked intentions. They make use of malicious programs (also known as malware) for this purpose.

A malware is a software program that achieves the damaging intent of an attacker [1]. According to NIST (National Institute of Standards and Technology), it is a program that has the intent of compromising the Confidentiality, Integrity, or Availability of the victim machine and its resources [2]. According to Internet Security Threat Report, Symantec [3], over 430 million new malicious specimens were discovered in the year 2015 which is about 36% more than that in 2014. Malware writers are making use of obfuscation techniques like insertion of dead code, subroutine reordering, instruction substitution etc. for creating polymorphic and metamorphic malware [4]. Moreover, the malware are becoming sophisticated, targeted, persistent, stealthy and unknown day by day. These have the capability to circumvent the earlier developed methods of detection and mitigation which clearly shows the need of shifting from traditional cyber security to Cyber Threat Intelligence (CTI). Popular security organizations providing CTI services include FireEye, LogRhythm, RSA, Symantec, and Verisign etc.

1.1. Cyber Threat Intelligence.

According to Gartner, "Threat intelligence is evidence-based knowledge, including context, mechanisms, indicators, implications and actionable advice, about an existing or emerging threat to assets that can be used to take decisions" [5]. It answers the questions like what methods are being used by attackers? What is their motive? What platform they are targeting at? etc. Secureworks [6] identifies CTI as a service, which is intended to help clients by providing them with early warnings on emerging threats, vulnerabilities and consultation with the threat intelligence group to have discussion on the same. Threat

*Department of Computer Science and Engineering, PEC University of Technology, Chandigarh, India and Department of Computer Science and Engineering, Chitkara University Institute of Engineering and Technology, Chitkara University, India (ekta.gandotra@gmail.com)

†Department of Computer Science and Engineering, PEC University of Technology, Chandigarh, India (divya@pec.ac.in)

‡Department of Computer Science and Engineering, PEC University of Technology, Chandigarh, India (sanjeevsofat@pec.ac.in)

intelligence doesn't only provide general threat information, but also aid in prediction of a particular threat to a specific organization in future. The security community requires the methods that have the capability to provide the situational awareness and would have the ability to proactively detect and respond to new malware threats. Therefore, it is imperative to obtain CTI about the evolving malware internally as well as from the external sources. The intelligent information so obtained can help the organizations and other stakeholders to focus and prioritize the defensive methods used to deal with future threats [7].

This paper proposes the design of a framework for generating CTI (specifically targeting the malware threat), which on the other hand can act as EWS. It provides the detailed description of the framework explaining all its components and various modules along with their functionality. It describes various tools and methodologies used for the framework design. The framework, when deployed in a cloud environment can analyze and distinguish malicious traffic from the benign one and can provide security-as-a-service. It also presents the real-time testing of the proposed framework which is realized by designing a prototype for providing security-as-a-service. This type of intelligence can help the security analysts to take the preventive measures to stop the future attacks or to minimize the risks posed by them.

2. Framework Design and Architecture. This section provides an overview of the design and architecture of the framework for generating malware threat intelligence. Investigation of the incidents or changes caused by a malicious program on the victim machine can definitely help in its detection and take preventive measures to avoid similar attacks in future. Table 2.1 briefly describes the steps involved in the process of malware detection and prediction which is the significant part of malware threat intelligence framework.

TABLE 2.1
Steps for generating malware threat intelligence

Step 1: Incident Response
Step 1.1 Data Capturing
Step 1.2 Malware Detection
Step 1.3 Incident Reporting
Step 2: Conduct In-depth and Conclusive Analysis
Step 2.1 Data Acquisition and Preparation
Step 2.2 Malware Analysis
Step 2.3 Malware Classification and Threat Assessment
Step 2.4 Visualization and Reporting

Each step is further comprised of several sub-steps. The detailed process is described by using a flow diagram shown in Fig. 2.1. The process begins by the definition of the ***Scope and Purpose*** of the task. The purpose of performing malware analysis could be to identify or detect zero-day malware and to get prepared to prevent their future occurrences and/or to identify the source of malware attack and to study the behavioral trends to take the preventive measures to stop such attacks in future or to minimize the risks posed by them.

In the ***Groundwork*** step, the sources from where data is to be captured/collected are identified. Proper permission and rights are obtained from the concerned authorities and administrators.

Data Acquisition is the step where network data is monitored to identify and capture malicious data or it is acquired from various Virus collection databases which are available for download in the public domain after free registration. The clean files can be acquired manually from System32 directories of Windows 2000, Windows 2003, Windows XP and Windows 7. After collecting random malicious samples from diverse sources, these are scanned using an Anti-Virus (AV) tool in the ***Data Preparation*** step to confirm them as malicious and then filtered to keep only the unique malware (based on their MD5 hashes). ***Automated Malware Analysis*** is performed by executing the acquired malicious and clean files in the sandboxed environment to produce the analysis reports which are preserved in the ***Data Preservation*** step. These reports are then studied and analyzed to identify useful features. The irrelevant features are removed in ***Feature Extraction*** step. ***Data Mining*** involving machine learning and statistical modeling is used for knowledge discovery in the data. Supervised machine learning algorithms are used in our present work to build the classification models

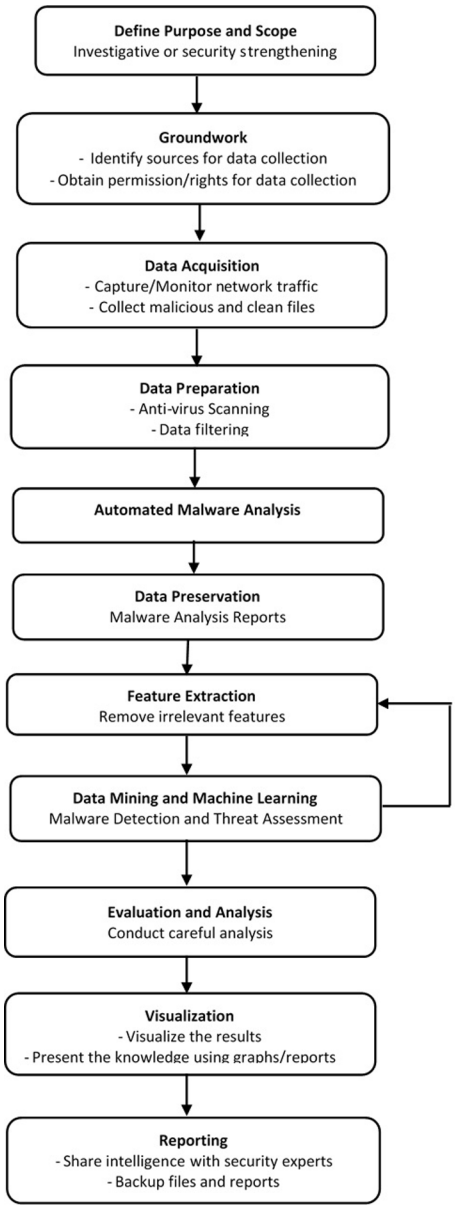


FIG. 2.1. Malware detection and prediction process.

for zero-day malware detection which are then compared to finalize the most suitable one. An unsupervised Fuzzy modeling is used to assess the threat level of a malware on the basis of which it could be prioritized for performing manual analysis. ***Evaluation and Analysis*** is the key step of the process. The machine learning algorithms are evaluated for their performance. The evaluation and analysis results are presented in ***Visualization*** step. Data visualization is the graphical representation of data which is helpful for obtaining an overall view and locating important aspects within the dataset. ***Reporting*** is the concluding step of the process in which the generated intelligent information is shared with security experts and other stakeholders. On the basis of this information, they issue early warnings and corresponding remedial actions to be taken to deal with emerging malware threats.

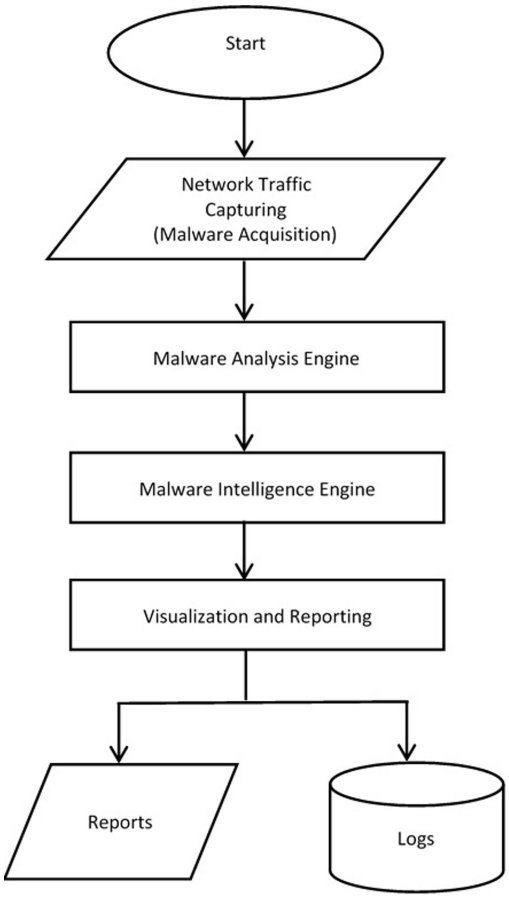


FIG. 2.2. Design overview.

2.1. Design. The proposed framework has been designed while keeping in mind the objectives like extensibility, scalability, quick incident reporting and efficiency. Fig. 2.2 shows an overview of the design of the proposed framework. Broadly the design can be classified into three phases. After acquiring the malware samples, these are scanned with AV engines in the first phase so as to filter out according to the family to which they belong or on the basis of the operating system they target. Thereafter, these are made to execute in the sandboxed environment to generate malware analysis reports. These reports are then analyzed to gain insight into the malware behavior. In the 2nd phase, pattern matching is done on the malware behavior data and classification model is built to detect zero-day malware and also their threat level is assessed so that these specimens could be prioritized for allocating resources in order to perform closer manual analysis. This phase raises alerts and creates logs containing information about the detected malware and their threat level. For the newly detected malware samples, signature generation is done and is stored in the database repository. For the next phase, malware behavioral trend analysis is performed and visualized to have an insight into the malware behavior. The information so obtained can be shared with security agencies and other stakeholders so that they can issue advisories and preventive measures to be taken to deal with future malware threats.

2.2. Architecture. This section gives the detailed description of architecture of the complete system. The proposed framework has the capability to provide a greater awareness of emerging threats and can deal with malicious programs in a proactive way. It combines the malware analysis data consisting of behavioral attributes, their classification into the existing families or identifying those which don't fall under any class, signature generation, malware detection, their behavioral trend analysis etc. The main components of the

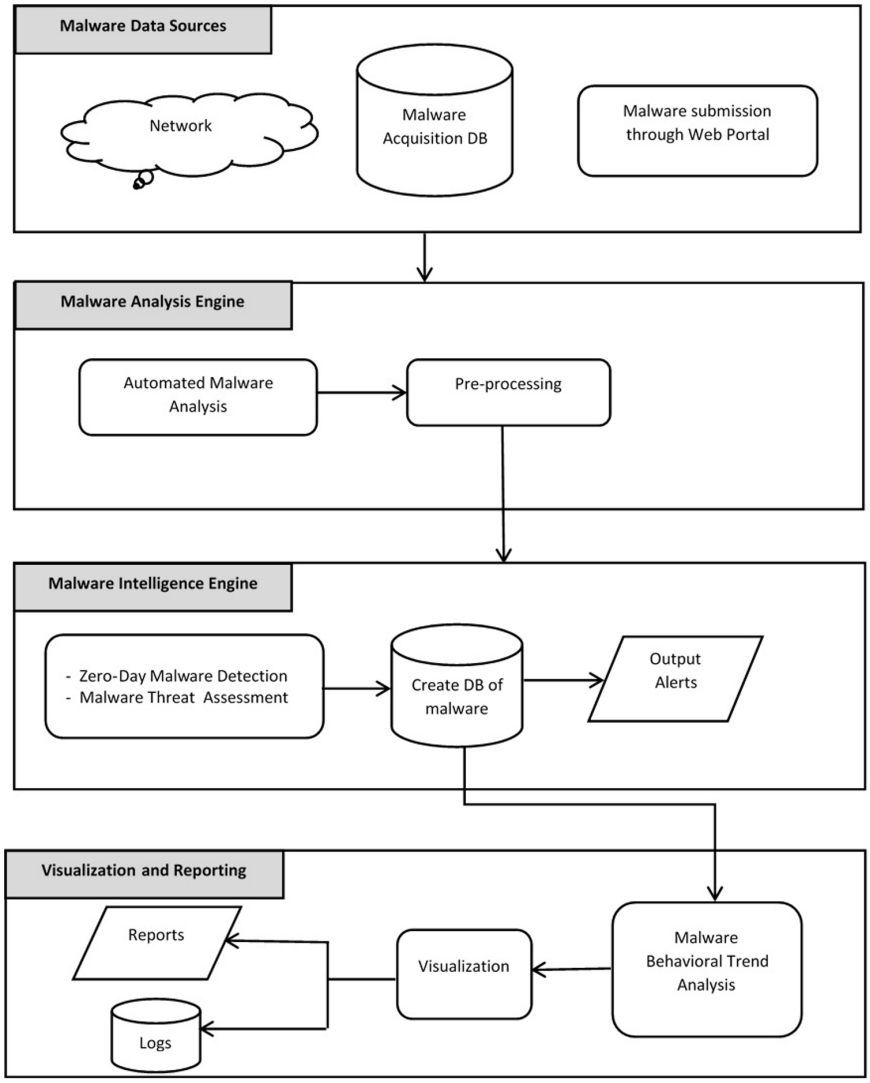


FIG. 2.3. Architecture of proposed framework.

architecture are shown in fig. 2.3 and are discussed in what follows.

2.2.1. Malware Data Sources. Random malware specimens can be acquired from various Virus collection repositories after free registration. These can also be collected by designing a web-portal where the users are permitted to submit suspicious/malicious samples. Another way to collect malicious files is to capture and monitor user's network traffic.

2.2.2. Malware Analysis Engine. Malware analysis is the skill of dissecting malicious programs for understanding their functionalities and potential impact on the victim machine. It is performed using two broad methods: Static malware analysis (code analysis) and dynamic malware analysis (behavioral analysis). Static malware analysis is performed by examining the binary file without executing it. Dynamic malware analysis, on the other hand, is performed by monitoring its behavior while it is being executed in the virtual environment [8]. Academicians and Industrialists working on malicious programs use these techniques to have an understanding of their functionalities & motives and thus the menace level posed by them. With the intention of resolving the challenges raised by the large volume of malware samples, automated malware analysis is

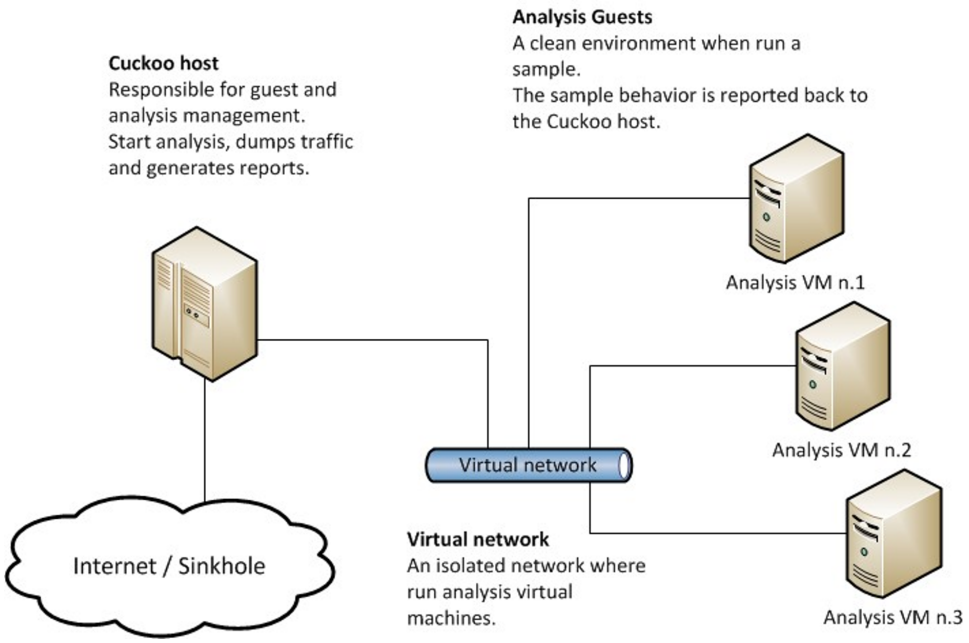


FIG. 2.4. Architecture of Cuckoo sandbox [9].

used. It automatically executes the malicious binaries in a sandboxed environment and generates the reports containing both static and dynamic features of malware programs. A comparison of tools and techniques for performing static, dynamic and automated malware analysis is carried out in [9]. It also provides the examples from the literature that use different malware analysis techniques for extracting their behavioral attributes and use machine learning algorithms for discriminating malicious programs from benign ones.

In order to develop the malware intelligence component, malicious binaries targeting Windows operating systems are acquired from Virusshare[1]. These specimens are then scanned using AVAST[2] AV and filtered to keep only the unique MD5 hash files. Subsequently, these are executed automatically (using a python script) in a sandboxed environment set up at Cyber Security Research Centre (CSRC), PEC University of Technology, Chandigarh, India. We used Brad Accuvant[3], a modified version of Cuckoo sandbox[4] for this purpose which offers numerous improvements over regular Cuckoo. It is an open source automated malware analysis system developed in python. It consists of central management software for handling the task of malware analysis. The main constituents of Cuckoo's architecture are: host machine and a number of guest machines as explained in [10]. The host machine runs the core component of the sandbox whose function is to manage the complete analysis process and the guest machines are the isolated environments where malware specimens get executed safely. Fig. 2.4 depicts the high level design of Cuckoo's architecture.

Cuckoo provides a series of Python scripts to hook into and examine malware activities while it is being executed. The system uses CuckooMon as the core element that provides it the capability to monitor a malicious sample by intercepting its execution flow. It is configured on a server with Ubuntu LTS 14.04 as the host machine and Windows XP, SP3 as the guest machine using Oracle VirtualBox[5]. It is available as open-source and has the ability to create multiple isolated virtual machines running different operating systems (shown in fig. 2.5). It has the ability to create current snapshot of complete virtual hard-disk that can be restored to its saved stage at any time. It also includes command line management interface which can perform everything that

[1] www.virusshare.com
[2] www.avast.com
[3] www.github.com/brad-accuvant
[4] www.cuckoosandbox.org
[5] www.virtualbox.org/

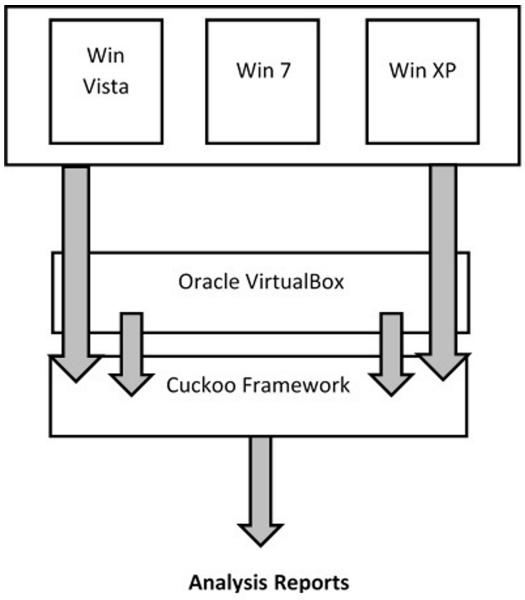


FIG. 2.5. VirtualBox as virtualization software.

can be done using Graphical User Interface (GUI). Due to all these functionalities, VirtualBox is used for the proposed system. It contains command line interface Vboxmanage, which provides the commands to manage virtual machine easily. For instance, to start and stop virtual machine and restore it to clean stage, which is very important while dealing with malware specimens.

After executing the files in the virtual environment, Cuckoo generates malware analysis reports in different formats like Java Script Object Notation (JSON), Hypertext Markup Language (HTML), Portable Document Format (PDF) etc. to make these user friendly. The reports generated are then parsed to obtain the various malware features such as MD5, file name, file size, timestamp of malware creation, packer used, suspicious section, dropped files, mutex creation, file system activities, registry activities, network activities and services created etc. These malware features are then used as a basis for generating malware intelligence.

2.2.3. Malware Intelligence Engine. Providing intelligence of the cybercrime and investigating complex threats is very crucial. Malware intelligence includes samples' analysis report, information regarding detection of new malware specimens, their signatures, behavioral profile and other informative knowledge about the malicious program which can be used for generating early warnings. The following techniques (as a part of this component) have been proposed for generating malware threat intelligence.

- **Zero-Day Malware Detection:** The traditional defenses like AV software and IDS/IPS typically rely on signature-based methods and are unable to detect previously unseen malware to offer immediate protection as these are needed to be analyzed before creating signatures for them. In order to solve this issue, static or dynamic malware analysis is being used along with machine learning algorithms for malware detection and classification [11][12], but the main problem with these systems is that they have high false positive and false negative rate. Another problem is that the process of building classification model takes time (due to large feature set) which hinders the early detection of malware. So the challenge is to select the relevant set of features so that the classification model can be built in less time with high accuracy. A survey conducted by Gandotra et al. [13] on malware analysis and classification clearly shows that a single approach, either static or dynamic can't classify the malware specimens accurately. So, a hybrid technique integrating both static attributes and dynamic behaviors is required for better malware detection and classification. After performing malware analysis, the obtained features are pre-processed and specific features are selected along with machine learning techniques to detect and classify zero-day malware. We proposed a malware classification system [14], which uses integration of

both static and dynamic features for distinguishing malware files from clean ones using machine learning algorithms with the aim of achieving excellent classification accuracy. The malware features which are used in this work include suspicious section count and function call frequency (static features) and file & network activities (dynamic features). This malware detection system is able to achieve an accuracy of 99.58% for the machine learning algorithm, Random Forest. In order to improve the classification model building time, we conducted another set of experiments [15] which considers the integration of both static and dynamic analysis features of malware binaries followed by a filter approach for selecting relevant set of features. Static and dynamic analysis features considered together provide high accuracy for distinguishing malware binaries from clean ones and the relevant feature selection process improves the model building time without compromising the accuracy of malware detection system.

- **Malware Threat Assessment:** Industries providing Anti-malware solutions need the skilled analysts and a large number of hardware resources to analyze the malicious programs more closely. The malware samples are increasing in volume day by day which makes it difficult to assign such a large number of resources for conducting a closer analysis. Thus, it is required to find a way to prioritize the newly discovered malicious programs for assigning such resources in order to conduct closer manual analysis. This component is designed to compute the threat or damage posed by a piece of malware (to a victim machine) automatically as soon as it appears in the wild [16]. The first step of this component involves a novel categorization of malware behaviors. This categorization is done using MAEC[6] (Malware Attribute Enumeration and Characterization) framework. The second step involves computing the impact of each behavior on the victim machine. It is done by assigning a weight to each behavior on the basis of its adverse effects on Confidentiality, Integrity and Availability of the target system and it's resources. The malicious programs use evasion techniques to evade their detection and to avoid the static and dynamic malware analysis. The evasion techniques used by each malware are reviewed to find their sophistication level. The third step involves designing a Fuzzy Inference System (FIS). We used Mamdani model [17], which is the most popularly used for representing human reasoning. It describes the association between fuzzy variables. The designed FIS takes malware behavioral impact and the sophistication level as the input variables and provide the threat level of a malware as the output variable. Based on these threat levels, the malicious programs can be prioritized for allocating resources for conducting a closer manual analysis.

2.2.4. Visualization and Reporting. The intelligence generated in the modules of above mentioned component is visualized and the reports generated regarding malware behavior are stored in the database repository. This behavioral information can be further analyzed to have an insight into the changing trends of malware behavior. We conducted a statistical trend analysis of behavioral attributes of about 0.1 million historical malware specimens collected from diverse sources [18]. We also highlighted the challenges evolving out of these trends and provided the future research directions to malware analysts and security researchers. The intelligent information or insight so obtained about malware threats can be shared with Computer Emergency Response Teams (CERTs), security agencies and other stakeholders near real-time and thus act as an early warning system. This information can be shared with the stakeholders using threat Information sharing Standards like Structured Threat Information Expression (STIX) [19], Trusted Automated Exchange of Indicator Information (TAXII) [20] and Cyber Observable Expression (CybOX) [21], which are the efforts to enable automated cyber threat information sharing across various security organizations and stakeholders.

3. Framework Realization. This section explains the integration of all the tools and methodologies used for testing the proposed framework. It is realized by designing a prototype capable of providing security-as-a-service. The Squid proxy server [7] has been used for capturing the network traffic while implementing the proposed framework. The complete workflow of the system has been divided into two halves: Online Mode and Offline Mode and is depicted in the fig. 3.1. The step by step description of both the modes along with their implementation is discusses as follows.

[6] https://maec.mitre.org/
[7] www.squid-cache.org

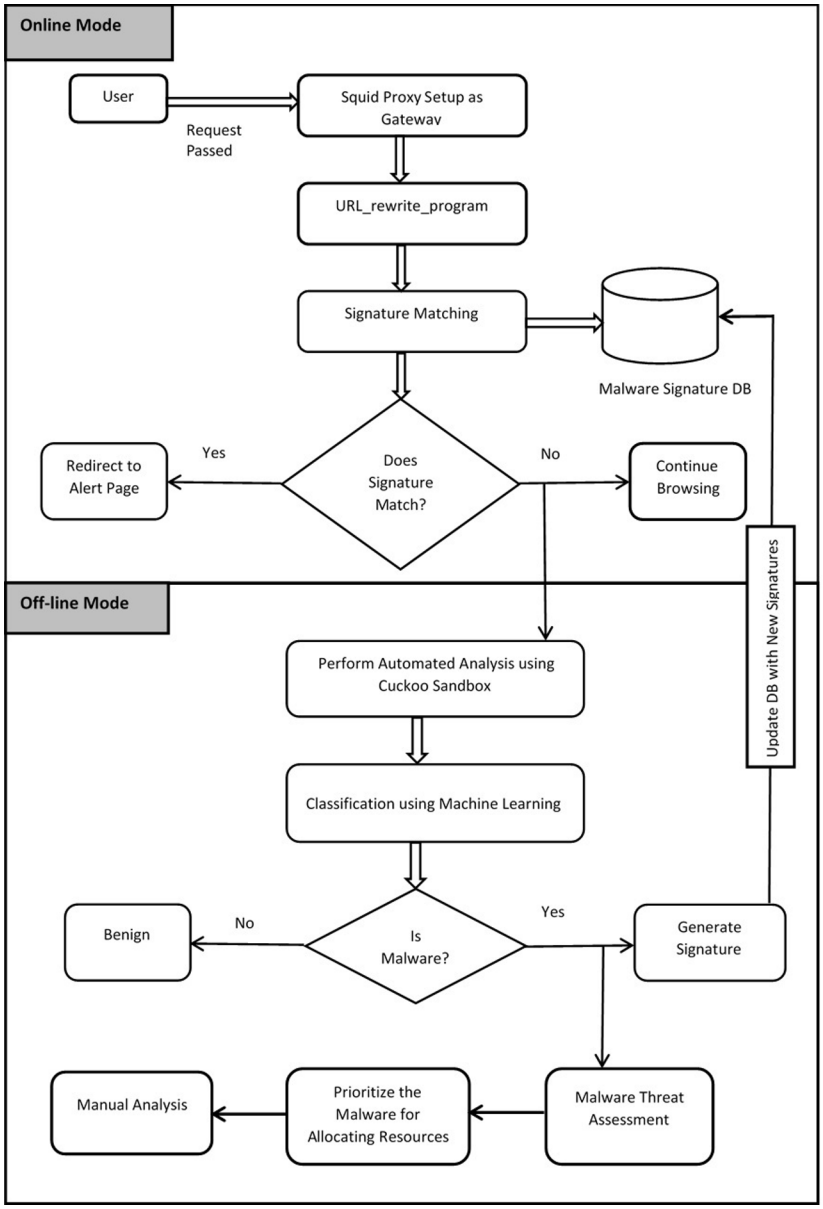


FIG. 3.1. System prototype for providing security-as-a-service.

Online Mode

The first step in the process is to capture the Internet traffic, especially the URLs the user is accessing. For this purpose, a Squid proxy server is installed in the transparent mode. The entire URL requests made by the users are captured by Squid, and forwarded to the url_rewrite_program to process the URL before request gets completed.

- After the url_rewrite_program receives the URL requested by the user, the complete content of that URL is fetched (including its css, JavaScript, images, .exe and everything else it contains). These fetched contents are then verified against set of viruses' signatures that are acquired from an open source AV engine, AVAST. If signature matches, user is redirected to an Alert Page (as explained in next step) else the user is allowed to continue browsing and for detecting zero-day malware in the suspicious traffic,

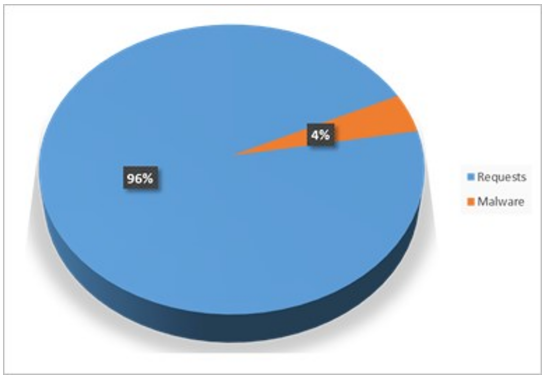


Fig. 3.2. Malicious requests against total requests.

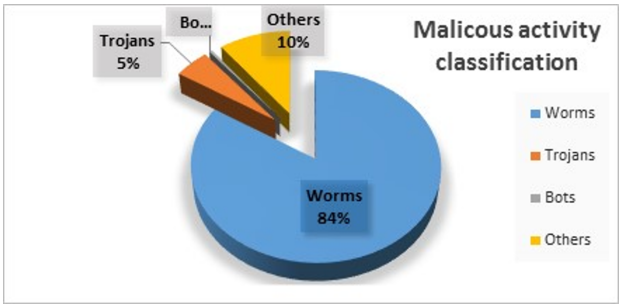


Fig. 3.3. Malware family classification.

the control goes to the offline mode.

- After it is found that the user is accessing some malicious content, instead of completing the request, user is warned by redirecting to alert page which has been hosted on the same system using Apache2 server and Python flask web development environment. It gives the details about what URL user tried to access, from which IP address and what was the kind of the malware.

Offline Mode

- In the offline mode, an automated analysis is performed for the suspicious files. In this process, every sample is made to execute automatically through a python script in a virtual environment created using Cuckoo sandbox and virtualbox which generates the comprehensive information in the form of malware analysis reports.
- These reports are then analyzed for selecting relevant set of features which are then used to predict whether the content is malicious or not using machine learning algorithms. If a new malware is detected at this stage, then its signature is generated and the existing malware signature database (in the online mode) is updated so that it can be detected in real-time next time.
- After classifying malware into different classes, the damage capability level of a malware is computed which can help in providing early warnings about it so that immediate attention could be paid to it in terms of allocating resources for performing closer analysis.

For real-time testing purpose, the system was deployed at the premises of STPI (Software Technology Parks of India), Mohali and real-time Internet traffic was routed through it for 2 hours. A total of 1,84,000 URL requests were routed through the system. These were analyzed and results were logged in a log file for further analysis and reporting purpose. It is found that out of all analyzed requests, 4% were detected to be malicious (fig. 3.2). Out of the total malicious activities, more than 80% were worms (fig. 3.3).

From the log files generated while monitoring network traffic, other type of analysis can be performed in terms of number of total malware detected, number of unknown malware detected, total number of attacks etc.

The changing trends of malware attack can be studied to have an idea about the future attacks and preventive measures to be taken to stop these attacks or to implement risk mitigation strategies.

4. Conclusion. In this era, the Internet consists of dynamic, virtualized and scalable resources which are provided as services. Due to the tremendous increase in these services over the Internet, an easy way opens up for attackers to carry out malicious activities. As a result the end-users have to deal with a flux of new malware threats, which are persistent, stealthy and advanced. The traditional anti-malware solutions are not able to deal with the current threat landscape. So, there is a need to shift from traditional cyber security to cyber threat intelligence. The intelligence must be collected from multiple sources, analyzed efficiently and shared in standard formats within no time so that it can be leveraged and included into security systems for taking preventive measures. This paper discussed the high level design and the detailed architecture of the framework for generating malware threat intelligence. The intelligent information generated can be shared with security agencies so that they can issue advisories and preventive measures to deal with future malware threats.

REFERENCES

[1] U. Bayer, A. Moser, C. Krugel, and E. Kirda, *Dynamic analysis of malicious code*, Journal of Computer Virology 2 (2006), pp. 67-77.

[2] P. Mell, K. Kent, J. Nusbaum, *Guide to Malware Incident Prevention and Handling*, National Institute of Standards and Technology (NIST), 2005.

[3] Symantec, *Internet Security Threat Report* , 21 (2016), available at: https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf

[4] S. Singla, E. Gandotra, D.Bansal, and S. Sofat,*Detecting and Classifying Morphed Malwares: A Survey*, International Journal of Computer Applications, 122 (2015).

[5] Webroot, *Threat Intelligence: What is it, and How Can it Protect You from Today's Advanced Cyber-Attacks*, available at: https://www.gartner.com/imagesrv/media-products/pdf/webroot/issue1_webroot.pdf.

[6] SecureWorks, *Threat Intelligence*, available at: https://www.secureworks.com/capabilities/threat-intelligence.

[7] E. Gandotra, D. Bansal, S. Sofat, *Computational techniques for predicting cyber threats*, In Intelligent Computing, Communication and Devices, Springer (2015), pp. 247-253.

[8] M. Sikorski, and A. Honig, *Practical malware analysis: the hands-on guide to dissecting malicious software*, No starch press (2012).

[9] E. Gandotra, D. Bansal, and S. Sofat, *Tools & Techniques for Malware Analysis and Classification*, International Journal of Next-Generation Computing, 7 (2016), pp.176-197.

[10] *Cuckoo Sandbox Book*, available at: http://docs.cuckoosandbox.org/en/latest/.

[11] A.Saini, E. Gandotra, D. Bansal, and S. Sofat, *Classification of PE files using static analysis*, Proc. of 7th International Conference on Security of Information and Network, ACM, (2014).

[12] S. Singla, E. Gandotra, D.Bansal, and S. Sofat,*A Novel Approach to Malware Detection using Static Classification*, International Journal of Computer Science and Information Security, 13 (2015).

[13] E. Gandotra, D. Bansal, and S. Sofat,*Malware Analysis and Classification: A Survey*, Journal of Information Security, 5(2014), pp. 56-65.

[14] E. Gandotra, D. Bansal, S. Sofat, *Integrated Framework for Classification of Malwares*, Proc. of 7th International Conference on Security of Information and Network, ACM, (2014).

[15] E. Gandotra, D. Bansal, S. Sofat, in press *Zero-Day Malware Detection*, Proc. of 6th international symposium on embedded computing and system design, IEEE, IIT Patna, India, (2016).

[16] E. Gandotra, D. Bansal, S. Sofat, *Malware Threat Assessment Using Fuzzy Logic Paradigm*, Cybernetics and Systems, 48(2016), pp. 29-48.

[17] E. H. Mamdani and S. Assilian, *An experiment in linguistic synthesis with a fuzzy logic controller*, International Journal of Man-Machine Studies, 7(1975), pp. 1-13.

[18] E. Gandotra, D. Bansal, S. Sofat, in press *Malware intelligence: Beyond malware analysis*, International Journal of Advanced Intelligence Paradigms.

[19] S. Barnum, *Standardizing cyber threat intelligence information with the Structured Threat Information eXpression (STIX)*, The MITRE Corporation,(2012).

[20] J. Connolly, M. Davidson, M. Richard, M., and C. Skorupka, *The trusted automated exchange of indicator information (taxii)*, The MITRE Corporation, 2013.

[21] E. Casey, G. Back, G., and S. Barnum, *Leveraging CybOX to standardize representation and exchange of digital forensic information Digital Investigation*,Digital Investigation, 12(2015), pp. S102-S11012.

Edited by: Kavita Sharma
Received: Nov 12, 2016
Accepted: Jul 3, 2017

DOI 10.12694/scpe.v18i3.1301
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 18, Number 3, pp. 207–218. http://www.scpe.org

AN ENERGY EFFICIENT AND TRUST AWARE FRAMEWORK FOR SECURE ROUTING IN LEACH FOR WIRELESS SENSOR NETWORKS

ARZOO MIGLANI*, TARUNPREET BHATIA†, GAURAV SHARMA‡, AND GULSHAN SHRIVASTAVA§

Abstract. Wireless Sensor Network (WSN) is an advanced technology and has been used widely in many applications such as health monitoring, environment monitoring, military purpose etc. Nature of this network is that they are often placed in an open environment and are susceptible to various attacks. Traditional cryptography methods are not supportable in WSNs as they have high energy and resource constraints. Trust management has been proved to be an effective measure to enhance security as well as to handle threats for WSNs. Trust can be defined as level of reliableness in a node. Low Energy Adaptive Clustering (LEACH) is a cluster based routing protocol for WSN which is superior to direct communication protocol and known for its minimum transmission energy. However, LEACH itself has some limitations related to security. In this paper, an energy efficient and trust aware framework for secure routing in LEACH (EETA-LEACH), has been proposed that improves LEACH protocol by introducing trust to provide secure routing, while maintaining originality of LEACH protocol. This approach is a combination of trust-based routing module and trust management module that works together to select trusted Cluster Head (CH). The simulation results demonstrate that proposed scheme is better in terms of network lifetime and Packet Delivery Ratio (PDR). It is verified that malicious nodes will not be selected as CH and trust value of a malicious node decreases with time.

Key words: WSNs: Wireless Sensor Networks, Trust Management, Cluster-Based WSNs, LEACH.

AMS subject classifications. 68M12

1. Introduction. Wireless sensor network is an infrastructure less network that consists of large number of sensor nodes distributed over an area to monitor and to collect a certain amount of data. However, WSNs are placed in an open environment and hence are susceptible to various kind of attacks. A node may get attacked by adversary and disrupt normal working of sensor network. Moreover, other security mechanism such as asymmetric cryptographic algorithm depends upon high computational capacity, power and resource [1, 2] and it is easy for an adversary to steal keys. Other mechanism such as authentication, message integrity, confidentiality has also being proposed to solve security issues but all these methods are only helpful in providing assurance towards outsider attacks. Therefore, trust management is an efficient solution for a compromised network and preventing internal attacks. Trust can be explained as a belief in reliableness of other node that is how much a node has confidence in establishing a secure communication with other node. Trust algorithm proposed in this paper is an extension of [3]. The motive of this paper is also same that is to remove flaws in LEACH by introducing secure cluster based selection. LEACH [4] is clustering based approach where nodes arrange themselves into local area called clusters. After every round a new cluster head is selected to balance energy load. Every new round starts with Set-up phase proceeded by Steady-state phase. In Set-up phase clusters are formed based on some threshold value which is given as below in Eq. 1.1.

$$T(n) = \left(\frac{p}{1 - p \times rmod(\frac{1}{p})} \right) \tag{1.1}$$

After this sensor nodes will decide which cluster head to choose based on signal strength. Next based on strength of sensor in a cluster, cluster head will form a TDMA schedule and each node is supposed to transmit within their allotted slot. After collecting data from every cluster members, CH aggregates data and then forwards data to BS.

But there are some problems associated with LEACH. Though cluster head selection procedure ensures that all nodes would get an equal chance to become cluster head but energy factor is not considered while selecting cluster head. After a time period, it is likely that a node with low energy may get selected as cluster head [5]. Another problem is that different cluster head elected would have different distance to base station,

*Department of Computer Science and Engineering, Thapar University, India (arzoomiglani4@gmail.com)
†Department of Computer Science and Engineering, Thapar University, India (tarunpreetbhatia@gmail.com)
‡Département d'Informatique, Université Libre de Bruxelles, Belgium (sharmagaurav86317@gmail.com)
§Department of Computer Science and Engineering, National Institute of Technology Patna, India (gulshanstv@gmail.com)

so their energy needs would also be different [6]. In addition, in large network, CH which is located at more distance from BS has to adopt multipath which consumes high energy. Moreover, energy demands for intra cluster communication are less than inter cluster communication. LEACH is completely dependent on cluster head for transmission and aggregation of data and a compromised cluster head would drop packets and may not perform task assigned to it thus making an unsecure network. Thus it is very important to choose a trusted CH. In this paper an improvement in LEACH protocol has been proposed, while maintaining the routing of original LEACH protocol.

In section 2, related work is discussed for enhancing lifetime of sensor network as well as some cluster based trust management schemes has been described. Proposed algorithm is presented in section 3. Section 4 presents simulation results followed by conclusion and future scope in section 5.

2. Related Work. Traditional methods of cryptography such as TinySec [7] can be used in prevention of external attacks in WSNs but they cannot block internal attacks. Watchdog and pathrater [8] were the initial work for enhancing security by means of trust. LEACH has inspired design of other cluster based protocol. As LEACH relies completely on selected cluster head for aggregation and transmission of data, so it is important to elect a reliable cluster head. Trust-based Low Energy Adaptive Clustering Hierarchy (T-LEACH) [9] designed by Song, Fei et al. proposed a trust based leach protocol that contains two components that is monitoring module and trust evaluation module. Routing part is same as original protocol; cluster head selection is dependent on trust value of cluster heads. With help of trust update slots new trust value are calculated and shared among cluster members. T-LEACH helps in less data loss than LEACH.

With a similar motive an improvement of LEACH Routing Protocol Based on Trust for Wireless Sensor Networks [10] (TM-LEACH) is proposed by Wang Weichao et al. where a cluster head adjusting procedure is introduced that create multipath with elected cluster head acting as routers. TM-LEACH ensures reliability of data transmission. Rather than relying on optimal solutions Dhulipala et al. proposed A Heuristic Approach Based Trust Worthy Architecture [11]. This model has taken mobility of nodes into account for better trust aggregation. Distributed trust is calculated within a cluster where every node calculates trust of every other node and centralized trust is calculated based on overall cluster performance. If more than 80% of nodes are trusted then cluster will be declared as a secure for communication There are many improvements in LEACH which has been proposed in literature, [12, 13, 14] proposed schemes for energy efficient LEACH, [15, 16] proposed work for better cluster head selection

CONFIDANT [17] is available with several versions, it is proposed with a trust manager along with a reputation system which analyzes the events described by the watchdog, then detecting and removing misbehaving nodes from network. J. Manickam. et al. [18] proposed another fuzzy based solution in 2014. This scheme has less overhead with respect to energy and memory consumption. Trust is calculated at both intra-cluster and inter-cluster level. Inside a cluster to reduce communication overhead only direct trust is calculated. At inter-cluster level both of direct as well as indirect trust is calculated. Direct trust is calculated using a sliding time window scheme. For calculating indirect trust recommendation are considered, for requesting a recommendation a trust request (TREQ) message is broadcasted to all neighbours whosoever comes in transmission range. For deciding final trust fuzzy if and then rules are applied to three parameters that are: direct trust, Recommendation inconsistency and number of fluctuations.

In 2011, Senthilkumar [19] et al. introduced Honey Bee Mating algorithm to select cluster head but security issues in cluster head selection were not considered. Sahoo et al.[20] in 2015 proposed a energy efficient cluster based model. The crux of the paper is to prevent malicious nodes of the network to be a cluster head. The proposed algorithm uses a light weight dynamic trust algorithm in addition with Honey Bee Mating algorithm to elect most eligible member as cluster head. To implement the routing part, LEACH is used as base protocol. Trust calculated by sensor nodes can be used by can be used by secure localization as proposed by Peng Li et al. [21] in 2017. The approach introduced has improvised security by identifying malicious nodes and detecting Sybil attack. In 2015 wei Luo et al. [22] proposed a scheme to detect internal as well as external attack. In order to detect external attack SHA-1 hashing algorithm is used, to differentiate normal nodes from noxious nodes and to detect internal attack trust values of nodes are used. In addition to this, to maintain confidentiality the message between nodes are encrypted using asymmetric or symmetric key cryptography. Several Routing Framework using key management protocols over past years. Secure trust based key management(STKF) [23]

is such a protocol for secure routing of data from source to destination using trust values of nodes. In this protocol, global trust value of routing path is computed by considering multiplication of trust values of nodes in route. Source node will select its authentic neighboring node to securely forward the data and neighbor selection will be done taking distance and trust relation between nodes.However,to avoid hacking of data and to maintain privacy data transmitted will be locked wit common pair of key. Another approach is proposed by yuxin liu et al.[24] to avoid black hole attack by using trust values. Source node will select its neighbour as next hop that has less distance with sink node and has trust values above the predefined threshold.

3. Proposed Model. For executing trust mechanism for sensor networks environment, following assumptions have been made:

- There are some malicious nodes present in the network
- BS has unlimited source of energy and it is free from any kind of attack
- If a node is performing some malicious activity then it will be penalized and its trust value will decrease.
- If a node is showing good behaviour, it will be rewarded and its trust value will be increased.
- Malicious nodes present in network are consuming more energy and dropping more packets than normal nodes.

Aim of this protocol is to choose trusted CH i.e. nodes with less trust value or less energy should not be selected as CH. Proposed work can be divide into two main modules that is trust management module and trust based routing module [9]. Figure 3.1 represents overall architecture of proposed algorithm.

- **Trust Management Module**: This module calculates trust based upon remaining energy, PDR and distance.
- **Trust-Based Routing Module**: It is almost same as basic LEACH protocols with some changes in it. Trust-based routing module uses trust management module to perform secure routing.

An improvement in LEACH protocol has been proposed, while maintaining the routing of original LEACH protocol. The scheme used to calculate trust is described below: Inputs

- Network area
- Number of nodes

Nodes will be randomly distributed in given area. Every node runs with an energy watcher, PDR calculator, distance estimator and trust supervisor [25, 26, 27]. Energy Watcher will calculate remaining energy of neighbour nodes and CHs, PDR calculator will calculate PDR of every node based upon number of packets dropped by node, Distance Manager will calculate and maintain distance between node and neighbours node along with CH distance with node.Trust Supervisor will maintain trust level of neighbouring nodes and CHs elected. For calculating trust value three factors will be considered that are remaining energy, PDR and distance i.e. nodes with high remaining energy, high PDR, and less distance between subject node and evaluated node will have more trust value and thus have high chances of becoming CH as compared to those nodes with low trust value, low PDR and high distance between nodes.

The four components of a node will work as follows:

- **Energy Watcher:** It will keep track of remaining energy of nodes. Energy model for the network is discussed as: To transmit a k-bit message with a distance of d, energy consumption can be calculated by Eq. 3.1:

$$E_t = E_e(k, d) + E_a(k, d) \tag{3.1}$$

  where E_t is the transmitting energy, E_e is energy required to run transmitter and receiver circuitry, Ea is transmitter amplifier energy and energy required to receive any packet can be calculated by Eq. 3.2:

$$E_r = k * E_e \tag{3.2}$$

  Hence energy will be consumed while transmitting or receiving packets in the network. As sensor networks are deployed in area where it is not possible to charge these nodes timely, so protocol designed should be energy efficient to save energy of these nodes and increasing network lifetime.
- **PDR Calculator:** This component will keep track of PDR. From the past records PDR calculator will maintain total number of packets sent to BS and how many of them are actually received by BS.

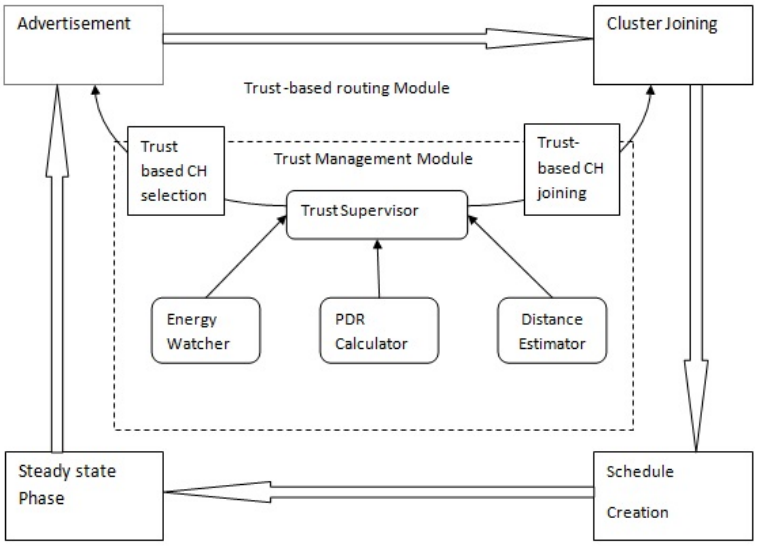


FIG. 3.1. *System Architecture*

Packets may be intentionally dropped by malicious node. Another reason for packets drop may be poor network connectivity. Nodes with high PDR will have high trust value and node with low PDR will have less trust value. Formula for PDR can be given by Eq. 3.3:

$$Packet\_Delivery\_Ratio = \frac{Packets\_Rcvd}{Packets\_To\_BS} \tag{3.3}$$

where Packets_Rcvd are total number of packets received by BS and Packets_To_BS are total number of packets sent to BS.

- **Distance Estimator:** This component will keep track of distance between nodes. If distance between evaluated node and subject node is less, a high trust value will be assigned to evaluated node otherwise if distance between subject node and evaluated node is high, then low trust value will be assigned to node. Hence trust value is inversely proportional to distance between nodes. Also this component will keep track of distance between nodes and CH.
- **Trust Supervisor:** This component will maintain trust values of nodes that will be used by routing module for trusted CH election and secure routing. The working of trust supervisor is being discussed in trust management module.

3.1. Trust Management Module. For calculating trust, trust supervisor will calculate both direct and indirect trust and final trust will be calculated by aggregating both trust values [28]. Direct trust is that trust which is calculated by nodes itself without scanning opinion of other nodes. Direct trust will be calculated based on past and present interactions of nodes. In order to save energy, sometimes it is not possible for a node to calculate direct trust of other nodes; in that case nodes will take recommendations from other nodes which will result in indirect trust. Indirect trust is also called second hand trust. In this model trust is calculated by considering energy, distance and PDR as trust metric. Nodes with high remaining energy, high PDR, less distance between nodes will have more trust value as compared to those nodes with less remaining energy, less PDR, more distance between nodes. As shown in Fig. 3.2 a subject node is one which wants to calculate trust of other node, evaluated node is one whose trust value is to be calculated, recommendation nodes are those whose opinions are considered for calculating indirect trust. Total trust of evaluated node will be computed by using both direct as well as indirect trust.

An initial trust of 0.5 is assigned to every node.The trust value of nodes can range between[0,1], where 1 represents that node is fully trustworthy, 0 represents complete distrust, 0.5 represents a normal trust value which

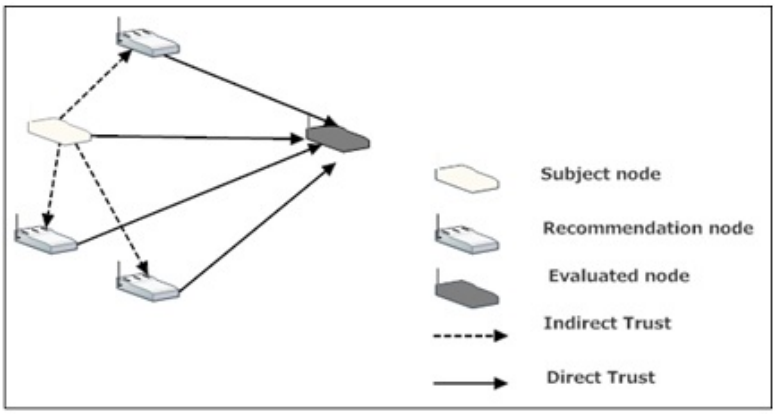


FIG. 3.2. *Trust Relationship*

can be ignored. For calculating direct trust, trust supervisor will interact with energy watcher, PDR calculator and distance estimator. For calculating direct trust, first trust supervisor will check remaining energy, and then a series of if-then rules will be applied to remaining energy, by comparing remaining energy with threshold value trust values will be assigned to nodes. Threshold values are selected by analyzing remaining energy after a particular round. Nodes will be awarded or penalized based upon the results after comparing remaining energy with threshold value. A node will be rewarded if its remaining energy is high after a particular round and if at the same round node is having less energy as compared to threshold then it will be penalized.

Once remaining energy has been checked, next trust supervisor will check PDR of nodes. PDR of nodes is compared with thresholds and then accordingly reward or penalty will be given. A node with high PDR will be rewarded and the nodes which drop more number of packets will have less PDR and hence penalized.

Further trust is dependent on another factor that is distance between nodes. If distance between nodes is high then corresponding trust of the node will be more and vice-versa. Hence direct trust can be calculated based upon aggregation of three factors.

Next, indirect trust will be calculated based on recommendations considered from other nodes. Indirect trust is the sum of trust values calculated by other nodes and given by Eq. 3.4:

$$IT^{C}_{A\to B} = \sum_{C} DT_{A\to C} \times DT_{C\to B} \tag{3.4}$$

where, $IT^{C}_{A\to B}$ is indirect trust of B calculated by A considering recommendation from C and C $\neq$ A; and are the direct trust value calculated by A for$C$ and C for B. For calculating final total trust both of direct and indirect trust will be aggregated as given below by Eq. 3.5:

$$TT_{A\to B} = wDT_{A\to B} + (1-w)IT^{C}_{A\to B} \tag{3.5}$$

$TT_{A\to B}$ is the total trust of node A on node B, w is the weight associated with direct and indirect trusts. A higher value of w signifies that sensor nodes relies more on its own judgment whereas a lower value of w signifies that sensor nodes has more trust on recommendations provided by other nodes. Final trust values of nodes will be stored by Trust Supervisor.

3.2. Trust-Based Routing Module. Routing module consists of two main phases: Set-up phase and Steady-state phase. In Set-up phase clusters are arranged and selected followed by .steady-state phase where nodes will transmit data to BS.

1. Set-up phase

a) Advertisement phase

This phase is same as in original LEACH protocol but for increasing lifetime of network energy factor is considered while selecting CH, so that the nodes with less energy should not get selected as CH. The number

TABLE 3.1
Neighbour CH Information

Attribute	Description
ID	ID of neighbouring CH
E_{REM}	Remaining energy of CH
T_{node}	Final trust of neighbouring CH
D_{Bs}	Minimum distance of neighbouring CH from BS
EC	How many times Neighbouring CH is elected as CH
$N_{nearest}$	Whether nearest neighbour or not

of nodes elected as CH with low energy will be less thus increasing network lifetime. To start procedure of CH election, node will select a random number between 0-1. If the number chosen is less than threshold value of node, node will be selected as CH otherwise not. The threshold value can be given by Eq. 3.6:

$$T(n) = \left(\frac{p}{1 - p \times rmod(\frac{1}{p})} \times \frac{E_{REM}}{E_{INT}}, n \in G \right). \tag{3.6}$$

where p is the desired percentage of CHs, G is set of nodes that have not been elected as cluster-heads in the last $1/p$ rounds and r is the current round, E_{REM} is remaining energy of node and E_{INT} is initial energy of node. After this phase, nodes has list of all eligible CH members.

Trusted CH Arranging Procedure

After CH has been elected, now elected CH will find all its CH neighbours [10, 29] and all information regarding CH neighbour will be collected from energy watcher, trust supervisor and distance manager. CH will maintain information of neighbour CH in form of a table. Each node will maintain an entry corresponding to every attribute mentioned in Table 3.1.

Now CH will examine whether neighbour is nearest neighbour or not and this will be decided by comparing distance of nodes with D. Equation 3.7 gives the value of D. Distance between CHs will be calculated with signal strength. If distance calculated is less than D then $N_{nearest} = 1$ else it is 0.

$$D = \text{å} \times \sqrt{\frac{1}{\pi K}} \times L \tag{3.7}$$

where L is the side length of the square area where sensor nodes are deployed, K is the number of cluster-heads, åis an adjusting factor. This will uniformly cover whole area CHs. If number of nearest neighbour CH is greater than 0 then CH will calculate trust weight associated with every nearest neighbour CH and trust weight, W_T is calculated by Eq. 3.8:

$$W_T = \alpha \times \frac{E_{REM}}{E_{INT}} + \beta \times \frac{T_{node}}{\sum T_{node}} + \gamma \times \frac{d(CH, BS)}{AD_{c\text{ß}BS}} + EC \tag{3.8}$$

where, α, β, γ are the weight factors selected accordingly. As for this paper energy is already considered as attribute for trust calculation, so for simulation a lower value of is considered. If some other attribute is selected then a higher value for α should be considered otherwise β can have higher value and EC is number of times node is selected as CH. T_{node} is the trust value of neighbouring CH obtained from trust management module. $\sum T_{node}$ is aggregation of trust of all nearest neighbour CH. CH with heaviest trust weight value is selected as new CH and will broadcast this information to other nodes and CH selected earlier will vanish. In addition, minimum distance of node from BS is also considered. CH distance to BS is compared with other nodes CH distance to BS and if difference between CH and BS is greater than predefined value, node will not be selected as CH. d(CH, BS) is distance calculated between CH and BS and $AD_{c\text{ß}BS}$ is the acceptable distance between CH and BS. Hence CH selected with this procedure will be trusted, with better energy and will help in saving energy as transmitting energy cost will be less.

b) Cluster Joining

In original protocol non-CH nodes join cluster based on signal strength received from CH but here nodes will select their CH based on trust values of cluster nodes.

c) Schedule Creation

CH receives all messages from nodes that would like to join cluster. Based upon strength of nodes in the cluster, CH begins to create a TDMA schedule and assign slots to non-cluster nodes to send data as well as to calculate trust value.

2. Steady State Phase

In steady state phase nodes will transmit sensed data to CH along with calculating trust. This phase can be divided into two slots data slots and trust slots [9] . After end of this phase every other round begins with set-up phase.

- **Data Slots:** Nodes will keep their transmitter on during their time slot only and will sense the data in the same time slot and send sensed data to CH selected meanwhile other nodes transmitters are off in order to save energy. It is assumed that CHs are having more energy than normal nodes so they keep their receivers always on to receive data from non-CH nodes of the cluster.
- **Trust Slots:** During this slot trust supervisor will calculate trust associated with their neighbors and CH based upon considered factors. Nodes update their trust value regularly. In addition, CH will calculate trust of neighbour CHs in this slot and updates their table.
  For communication within a cluster i.e. an intra-cluster communication, amplification energy must be less than a inter-cluster communication that is a communication between CH and BS [6]. The reason behind this is within a cluster distance between nodes is less, so less of energy is needed to transmit a message as compared to inter-cluster communication. Therefore more energy could be saved.

4. Simulation Results. The proposed algorithm EETA-LEACH has been designed in MATLAB [30]. It is considered that 100 nodes are randomly distributed over area of 100×100 m^2.

4.1. Selection of CH. Figure 4.1 shows random distribution of sensor nodes in an area of 100*100 sq. units and LEACH protocol is simulated for routing purpose. There are some malicious nodes present in the network. Malicious nodes are represented by a plus (+) sign, normal nodes are represented with a circle (o). In addition selection of CH in particular round is also presented in Fig 4.1, nodes that are selected as cluster head are represented with dark blue asterisk. It can be easily analyzed that if no security practises are adopted, malicious nodes present in network could be selected as CH. Hence as a result, malicious CH selected would drop packets received from cluster members witch in result reduce network performance.

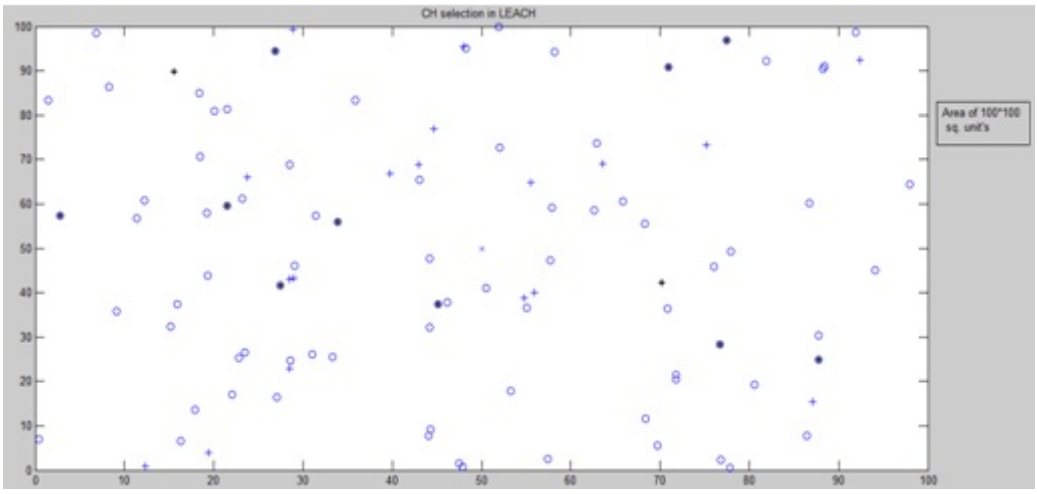


FIG. 4.1. *CH selection in LEACH*

After implementing EETA-LEACH, it is verified that chances of selecting malicious nodes as CH are almost negligible. In EETA-LEACH, CH is selected based upon trust values of nodes. Therefore selected CH will not be malicious. The whole scenario is represented in Fig 4.2. Trusted CH selected is represented by Green asterisk.

Algorithm 1 Trust Calculation Algorithm

procedure TRUST CALCULATION
input: Remaining energy, Packet_delievery_ratio,Distance between nodes
Every node is assigned with initial direct trust of 0.5
if(R.E >Th1)
DT= DT+5% of DT
elseif(Th2 <R.E <Th1)
DT=DT
elseif(R.E <Th2)
DT= DT-5% of DT
End
if(PDR >Th3)
DT=DT+5% of DT
elseif(Th4<PDR <Th3)
DT=DT
elseif(PDR <Th4)
DT= DT-5% of DT End
if(DS.N → E.N >Th5)
DT=DT+5% of DT
elseif(Th6 <DS.N → E.N <Th5)
DT=DT
elseif(DS.N → E.N <Th6)
DT= DT-5% of DT
end
Indirect trust will be calculated from recommendation nodes
TT= w *DT + (1-w)*IT
Notations:
DT= Direct Trust
IT= Indirect Trust
TT= Total Trust
Th=Threshold Value
DS.N→ E.N = Distance between subject node and evaluated node

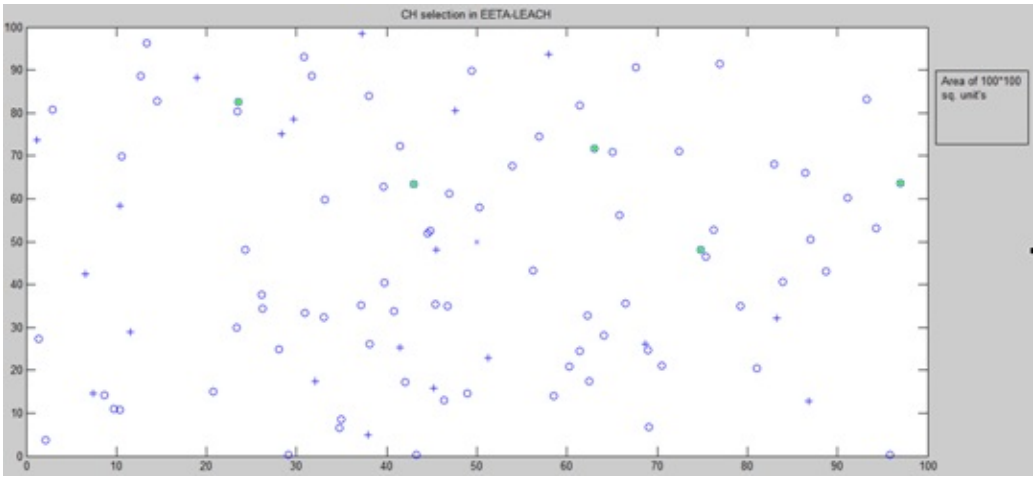


FIG. 4.2. *CH selection in EETA-LEACH*

4.2. Trust Evolution. Figure 4.3 plots trust value of a malicious node. Trust value of a malicious node decreases as time increases. The value of w1 and w2 is chosen to be 0.5 and 0.5 respectively in Eq. 3.5, which concludes that node is equally considering direct trust as well as recommended trust. At very first round node will have direct trust of 0.5, no indirect trust will be considered at first round, hence total trust will constitute to 0.5. Similarly at 10th round direct trust is 0.4800, indirect trust is 0.2438 and thus total trust is 0.3619 for this round. In the proposed model calculated trust is directly proportional to remaining energy and PDR, as malicious node consumes more energy, drops more packets therefore its trust value decreases as number of round increases.

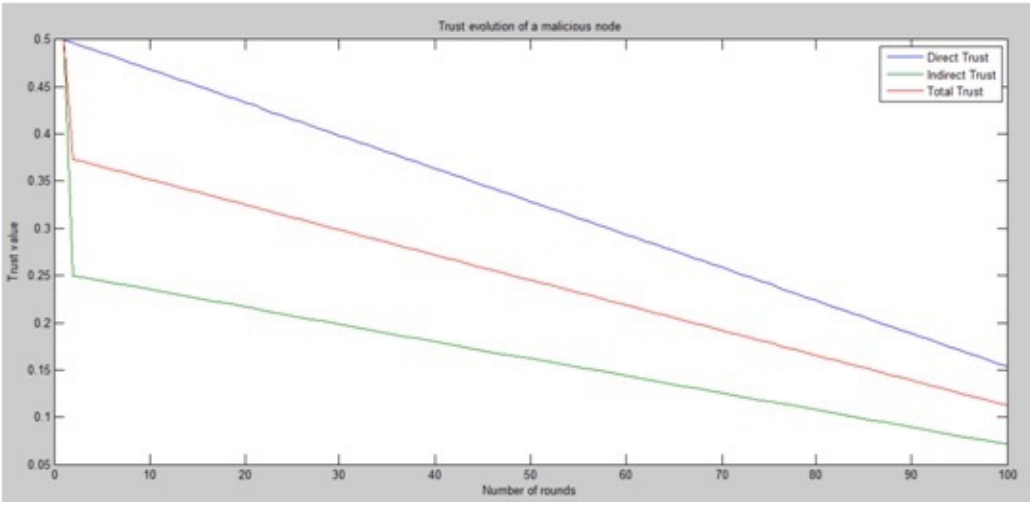


FIG. 4.3. *Trust evolution of a malicious node*

4.3. Analysis of Packet Delivery Ratio. Figure 4.4 shows number of malicious node versus average PDR. It could be observed that average PDR is 96% in EETA-LEACH and 91% in LEACH when there are no malicious node present in the network. There would be some packet loss because of poor network connectivity. Therefore PDR would not be 100% even if no malicious node present in the network. With presence of 5 malicious nodes in network EETA-LEACH network has PDR value of 0.9400 and LEACH has 0.8390, hence after implementing EETA-LEACH PDR increases by 12%. Similarly when 15 malicious nodes are present PDR is increased by 14% and with presence of 25 malicious nodes PDR increases by 21.5%. Hence it could be concluded that after implementing EETA-LEACH average PDR ratio is increased by 15.8%. EETA-LEACH has high average PDR as compared to leach because malicious nodes are not selected as CH and hence there is less packet drop in the network. Moreover EETA-LEACH can help in avoiding selective forwarding attack.

4.4. Network Lifetime Comparison. While comparing network lifetime it has been observed that EETA-LEACH has better lifetime when compared to LEACH as in LEACH there are many retransmissions as compared to EETA-LEACH. In addition in EETA-LEACH less of malicious nodes would be selected as CH so less consumption of energy as it is assumed that malicious nodes are consuming more energy. Moreover, consumption of less energy while intra-cluster communication as compared to inter-cluster communication and consideration of energy factor while selecting CH makes EETA-LEACH more energy efficient. It could be verified from figure 4.5 that in LEACH first node dies near 700th rounds as compared to EETA-LEACH where first node dies at 1100th round. Figure 4.5 shows network lifetime comparison.

5. Conclusion and Future Scope. In this paper an energy efficient trust based approach has been proposed which is combination of trust-based routing module and trust management module. In trust management module, trust supervisor calculates trust for nodes as well CH that can be used for trusted CH selection and secure routing. Total trust value is a combination of direct trust that is calculated by node itself and indirect

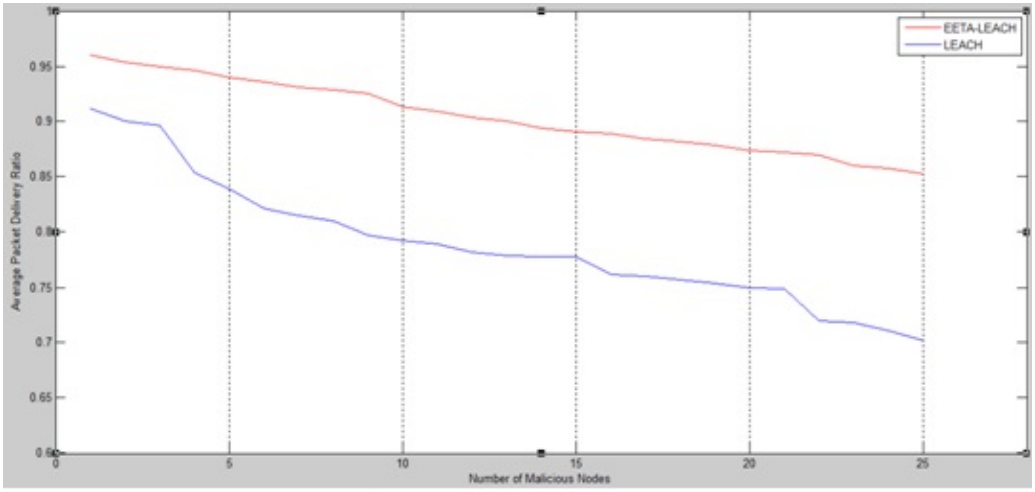


FIG. 4.4. *Packet Delivery Ratio versus number of malicious node*

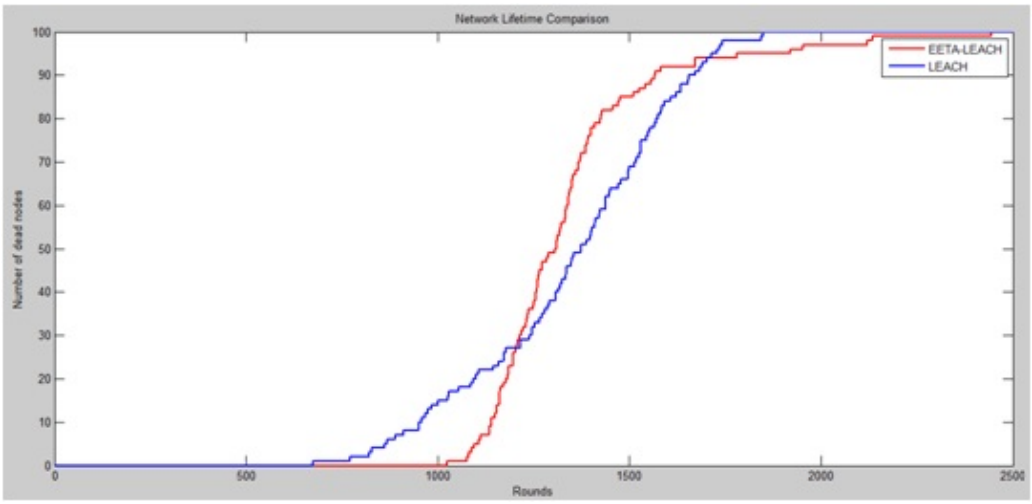


FIG. 4.5. *Network Lifetime Comparison*

trust which is trust from recommendation nodes. Trust-based routing module modifies original LEACH. Routing module comprises of further four phases that are advertisement phase, cluster joining, schedule creation and steady state phase. Nodes that are malicious in nature will have less PDR and consumes more energy. So these malicious nodes will not be selected as CH because their computed trust value will be less. In addition, routing module uses less energy for intra-cluster communication as compared to inter-cluster communication which would help in improving network lifetime. Protocol performance is verified using MATLAB simulator. It is verified that malicious nodes will not be selected as CH and trust value of a malicious node decreases with time. Simulation results proved that proposed algorithm consumes less energy and improves PDR as there are less number of retransmission. Average PDR is improved by 15.8%. In addition with implementation of EETA-LEACH, network lifetime improves as first node dies at 1100th round in EETA-LEACH as compared to LEACH where first node dies at 700th round. In future, this work could be extended by considering other types of WSNs e.g. dynamic WSNs, heterogeneous WSNs. Other social trust attributes such as privacy, intimacy, number of interaction could be considered in future to extend this work. Beside this trust model could be further extended to develop lightweight algorithm to further reduce energy consumption. Memory requirement for this

protocol are bit high as past records has to be stored by energy watcher, trust supervisor, PDR calculator. So challenging problem is to reduce memory requirements. In this paper, nodes that have less PDR are considered as malicious without considering the fact that less PDR could be less because of poor network connectivity, this also motivates future work.

REFERENCES

[1] Garcia-Luna-Aceves, J. J., and Spohn, M., *Source-tree routing in wireless networks*, In Network Protocols, 1999.(ICNP'99) Proceedings. Seventh International Conference on, pp. 273-282. IEEE, 1999.

[2] Cordasco, Jared, and Susanne Wetze, *Cryptographic versus trust-based methods for MANET routing security*, Electronic Notes in Theoretical Computer Science 197.2 (2008): 131-140.

[3] A. Miglani, Tarunpreet Bhatia, and Shivani Goel , *TRUST based energy efficient routing in LEACH for wireless sensor network*, In Communication Technologies (GCCT), 2015 Global Conference on, pp. 361-365. IEEE, 2015.

[4] Heinzelman, Wendi Rabiner, Anantha Chandrakasan, and Hari Balakrishnan , *Energy-efficient communication protocol for wireless microsensor networks* , In System sciences, 2000. Proceedings of the 33rd annual Hawaii international conference on, pp. 10-pp. IEEE, 2000.

[5] Naregal, Keerti, and Anand Gudnavar , *Improved cluster routing protocol for wireless sensor network through simplification* , In Advanced Computing and Communications (ADCOM), 2012 18th Annual International Conference on, pp. 1-3. IEEE, 2012.

[6] Mahmood, Danish, Nadeem Javaid, Sajjad Mahmood, Shaima Qureshi, Atif M. Memon, and Tariq Zaman , *MODLEACH: a variant of LEACH for WSNs* , In Broadband and Wireless Computing, Communication and Applications (BWCCA), 2013 Eighth International Conference on, pp. 158-163. IEEE, 2013.

[7] Karlof, Chris, Naveen Sastry, and David Wagner , *TinySec: a link layer security architecture for wireless sensor networks* , In Proceedings of the 2nd international conference on Embedded networked sensor systems, pp. 162-175. ACM, 2004.

[8] Marti, Sergio, Thomas J. Giuli, Kevin Lai, and Mary Baker , *Mitigating routing misbehavior in mobile ad hoc networks* , In Proceedings of the 6th annual international conference on Mobile computing and networking, pp. 255-265. ACM, 2000.

[9] Song, Fei, and Baohua Zhao, *Trust-based LEACH protocol for wireless sensor networks* ,In Future Generation Communication and Networking, 2008. FGCN'08. Second International Conference on, vol. 1, pp. 202-207. IEEE, 2008.

[10] Wang, Weichao, Fei Du, and Qijian Xu , *An improvement of LEACH routing protocol based on trust for wireless sensor networks* , In Wireless Communications, Networking and Mobile Computing, 2009. WiCom'09. 5th International Conference on, pp. 1-4. IEEE, 2009.

[11] Dhulipala, V. R. Sarma, N. Karthik, and R. M. Chandrasekaran , *A novel heuristic approach based trust worthy architecture for wireless sensor networks*, Wireless personal communications (2013): 1-17.

[12] Kim, Jin-Mook, Hyeon-Kyu Joo, Seong-Sik Hong, Woo-Hyun Ahn, and Hwang-Bin Ryou *An efficient clustering scheme through estimate in centralized hierarchical routing protocol*, In Hybrid Information Technology, 2006. ICHIT'06. International Conference on, vol. 2, pp. 145-152. IEEE, 2006.

[13] Chang, Ruay-Shiung, and Chia-Jou Kuo *An energy efficient routing mechanism for wireless sensor networks* , In Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on, vol. 2, pp. 5-pp. IEEE, 2006.

[14] Xiangning, Fan, and Song Yulin *Improvement on LEACH protocol of wireless sensor network* In Sensor Technologies and Applications, 2007. SensorComm 2007. International Conference on, pp. 260-264. IEEE, 2007.

[15] Lijuan, Yu, and Li Simin *Design and simulation of an improved LEACH algorithm in wireless sensor networks* ,Study of Optical Communications 5 (2008): 67-70.

[16] WU, Chun-tao, and Yan-jun HU *Improvement of LEACH in wireless sensor networks* ,Computer Technology and Development 3 (2009): 023.

[17] Buchegger, Sonja, and Jean-Yves Le Boudec, *Performance analysis of the CONFIDANT protocol*, In Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking and computing, pp. 226-236. ACM, 2002.

[18] Anita, X., M. A. Bhagyaveni, and J. Manickam, *Fuzzy-based trust prediction model for routing in WSNs*, The Scientific World Journal 2014 (2014).

[19] Senthilkumar, J., M. Chandrasekaran, Y. Suresh, S. Arumugam, and V. Mohanraj *Advertisement timeout driven bee's mating approach to maintain fair energy level in sensor networks*, Applied Soft Computing 11, no. 5 (2011): 4029-4035.

[20] Sahoo, Rashmi Ranjan, Abdur Rahaman Sardar, Moutushi Singh, Sudhabindu Ray, and Subir Kumar Sarkar *A bio inspired and trust based approach for clustering in WSN* , Natural Computing 15, no. 3 (2016): 423-434.Natural Computing 15, no. 3 (2016): 423-434.

[21] Li, Peng, Xiaotian Yu, He Xu, Jiewei Qian, Lu Dong, and Huqing Nie *Research on Secure Localization Model Based on Trust Valuation in Wireless Sensor Networks* , Security and Communication Networks 2017 (2017).

[22] Luo, Wei, Wenping Ma, and Qiang Gao *A dynamic trust management system for wireless sensor networks* , Security and Communication Networks 9, no. 7 (2016): 613-621.

[23] Kaur, Jugminder, Sandeep S. Gill, and Balwinder S. Dhaliwal *Secure trust based key management routing framework for wireless sensor networks*, Journal of Engineering 2016 (2016).

[24] Liu, Yuxin, Mianxiong Dong, Kaoru Ota, and Anfeng Liu *ActiveTrust: secure and trustable routing in wireless sensor*

networks , IEEE Transactions on Information Forensics and Security 11, no. 9 (2016): 2013-2027.

[25] Zhan, Guoxing, Weisong Shi, and Julia Deng , *Design and implementation of TARF: A trust-aware routing framework for WSNs* ,IEEE Transactions on dependable and secure computing 9, no. 2 (2012): 184-197.

[26] Raje, Radhika A., and Apeksha V. Sakhare , *Routing in wireless sensor network using fuzzy based trust model* , In Communication Systems and Network Technologies (CSNT), 2014 Fourth International Conference on, pp. 529-532. IEEE, 2014.

[27] Sakthidevi, I., and E. Srievidhyajanani , *Secured fuzzy based routing framework for dynamic wireless sensor networks* , In Circuits, Power and Computing Technologies (ICCPCT), 2013 International Conference on, pp. 1041-1046. IEEE, 2013.

[28] Fei, Xiong, and Xu Qi Jian *Active trust transmission mechanism for wireless sensor network*, In Intelligent Information Technology Application, 2008. IITA'08. Second International Symposium on, vol. 1, pp. 626-632. IEEE, 2008.

[29] Zhang, Y. *On The Improvement In LEACH Protocol of Wireless Sensor Networks*, Micro Computer Information, 2006,Vol.22(4-1), pp.183-185.

[30] `https://www.mathworks.in/products/matlab`.

Edited by: Kavita Sharma
Received: May 25, 2017
Accepted: Aug 2, 2017

DOI 10.12694/scpe.v18i3.1302
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 18, Number 3, pp. 219–228. http://www.scpe.org

DETECTION OF SARCASM IN TEXT DATA USING DEEP CONVOLUTIONAL NEURAL NETWORKS *

PULKIT MEHNDIRATTA, SHELLY SACHDEVA, AND DEVPRIYA SONI †

Abstract. With the ever increasing acceptance of Online Social Networks (OSNs), a new dimension has evolved for communication amongst humans. People now share continuous streams of messages to contribute their interests, indications, discuss activities, status, latest trends and much more. This gives an opportunity to monitor and mine the opinions of a large number of online active population in real time. On one side where researchers can find out a pattern to judge the mood of the user, a serious problem of detection of irony and sarcasm in textual data poses threat to the accuracy of the techniques evolved till date. In this paper, we are proposing a deep learning based convolutional neural network method that focuses on the skip gram technique to convert words to vectors and further to detect and identify the sarcasm and irony from the normal polarized text taken from Twitter[1]. We try to discover patterns from the textual datasets based on the identifiable features. Our approach is giving an overall accuracy of 89.9%.

Key words: Online Social Networks (OSN), Sarcasm, Irony, Automation Techniques, Sentiment Analysis, Opinion Mining, Deep Learning, Convolutional Neural Networks (CNN).

AMS subject classifications. 91D30, 92B20

1. Introduction.

The beginning of web 2.0 and Online Social Networking (OSNs) sites have given all new dimension to the world of communication among each other and has provided ample of opportunity for taking general public opinion to countable and provable patterns. Thus, these networks can be used as an effective means to testify trends and popularity in topics ranging from entertainment, politics, environment, economic or social issues and much more. Not, only people use standard languages like English, Spanish, German etc. but they try to be innovative by using emotion icons also known as emoticons, hashtags#, URLs and many more new things. People try to express themselves as much as possible whether they are happy or sad, annoyed or frustrated, excited or anxious, one thing they do not miss is a status update or a tweet.

These Online Social Networks like Facebook[2], Twitter, Linkedin, are the virtual worlds or one can say the second life to the people nowadays. Opinion mining is the sub-disciplines within the field of computational linguistics and natural language processing which refers to the mining, classification and understanding the mood and opinion of the people expressed in social networks, blogs, news forums and other user generated content. Analysis of the sentiment is often used while performing opinion mining to identify the sentiment, subjectivity, and other affecting states in user generated text. Around the globe researchers are analyzing trends and tracking user activity to predict the big question "What's next???". Not only this they are trying to see what people like and what they hate. They are trying to find out patterns in user search queries, browsing activity logs and much more. But one thing which from decades troubling the researchers in field of text and sentiment analysis is usage of Irony and Sarcasm.

Sarcasm is a well known, commonly used and well-studied topic in linguistics. In spite of being so widely used and being part of our speech, it's inherently very challenging not only for machines but for humans also to detect sarcasm in text. As the size of text communication is getting shorter day by day, this problem of identification of sarcasm poses a real threat not only to the efficiency of the machine learning algorithms but it's a great threat to the ultimate sentiment of the sentence where it is used. So, if not crucial but it is very important to address the problem of sarcasm in textual data for the further evolution and refinement of the various systems used for analysis of the sentiments. Some studies suggest, the problem with sarcasm is transformation of the overall polarity of the sentence apparently positive or negative to its opposite or negation. The rest of the paper is as follows: section II talks about the basics of sentiment analysis and opinion mining, majority focusing on the current state of the art. In section III, we discuss the overall status of the detection

*This work was supported by Department of Computer Science, Jaypee Institute of Information Technology, Noida, INDIA.
†Department of Computer Science and Information Technology, Jaypee Institute of Information Technology, Noida, INDIA({pulkit.mehndiratta, shelly.sachdeva, devpriya.soni}@jiit.ac.in).
[1]https://www.twitter.com
[2]https://www.facebook.com

of the sarcasm in textual data. The introduction to convolutional neural networks (CNN) is discussed under section IV. Here we have given a very brief introduction to what CNN's are and how they work. Section V discusses our methodology, where we explained the steps taken in this study for the detection of sarcasm in text. Section VI provides us with the results we achieved using CNN's for the detection of sarcasm, hence a fusion of text analysis and evolutionary algorithm working in tandem. Conclusion and future scope of this work has been discussed in section VII.

2. Sentiment Analysis and Opinion Mining. One thing that every person wants know that "What other people think" to help them in a decision making process. Many years before when we were at midst of World Wide Web, we asked our friends for their recommendations on various topics like clothing, whom they are planning to vote and much more. But with the help of Internet and technology at our disposal, it has become a matter of mere asking to find out the opinions and experiences of the vast pool of people who are neither our friends or acquaintance nor they are acclaimed professional critics - they are the ones we have never heard of. On the contrary, the number of people giving their views and opinion on the Internet is increasing day by day. According to a survey conducted in 2008 [1], almost 80% of the Internet population does online research before buying any product at least once. Not to mention, but people do research about restaurants, hotels, places to visit, various services (doctors, travels etc.) all the time, and this review system actually has a significant impact on the purchase pattern of the user. We show urgency to share that goods and services consumption is not the only motivation behind general user turning to seek online opinions and reviews. Gathering political information and update is another trending topic that people usually do online. Recent studies [2, 3] show that how a microblogging site can predict the results of the elections of various democracies like Germany, India and much more. This hunger and dependency of the user upon the online review systems and recommendations reveals one thing that people take interest in these new systems and take opinions of various users as a first-hand object. In [1], where authors claim that maximum people use and have positive experiences during online product research, around 60% of the population still reports that either the information about the products they are searching is missing, confusing or impossible to find. Similarly, in the case of elections result prediction [4], authors mention the shortcomings while examining the predictive power of social media to display conflicting results. Hence, there is a clear need to build better information systems than the systems which already exist.

2.1. Applications of Sentiment and Opinion Mining. Sentiments and opinions are central and integral part of almost all human activities. Whenever a decision is to be made by us we always look out for others' opinions and judgment. Businessmen, multi-billion dollars companies, or a naive user everyone needs opinions and he/she would ask family and friends. But with the overwhelming growth of online social networks, individual people and organizations are increasingly using the content for decision-making purposes.

2.2. Challenges in performing Sentiment and Opinion Analysis. Most of the techniques we studied and reviewed suggested that there is a bag of words known as sentiment lexicon which helps and used to express the positive and negative sentiment. Over many years in the past, researchers have designed algorithms and proposed ways to compile a lexicon for the very same purpose of sentiment analysis. For example, `good, great, better, nice, wonderful, amazing` are positive words and `bad, no, not, poor, worse, terrible` is set some negative words.

These words worth a great deal to many but they are way far from sufficient to perform sentiment analysis. Such lexicons are an integral part of the solution but they are not sufficient enough to perform the sentiment analysis. Some of the key issues we faced understanding sentiment analysis are as:

1. Sarcasm is one of the most common problems faced by many researchers, it's not only to find the sentiment rather say if the sentence is sarcastic or not e.g. *"I enjoyed my cab ride by paying $107 for 2 miles, thanks Uber :("*. This practice is more common among political reviews and opinions as compared to products and services.
2. The usage of sentiment lexicon words always make it easy for the automated detection of the sentiment of the sentence. But sometimes, the phrase or sentence may not have any word from the lexicon arising issues of correct classification. Most of the current systems would fail to classify them in correct bucket.
3. The exact orientation of the word used in the sentence may have opposite meaning and this depends on the domain to domain. Taking these things into account poses serious threat to the overall accuracy

and efficiency of the systems. Addressing all these problems need serious effort and dedication from the community working in this area.

3. Sarcasm and Irony. It all started when human invented language to communicate with each other, the way they discuss things, how they evolve solutions and also the art to amuse or mock others. Sarcasm is as old as language itself. Although its occurrence is not limited to English it is inherited part of British culture. For facts, BBC[3] has its own page to teach non-native English speakers the art of using sarcasm in conversation.

According to Oxford English Dictionary sarcasm means *a sharp, cutting expression or a remark, a taunt or bitter gibe.* Nowadays, people use sarcasm *when they want to say something opposite to the truth, in order to be funny, or just to make a point none the less.* On the other hand, irony is considered to be a special kind of sarcasm where users' are just want to convey the opposite to the literal interpretation.

Starting from the known work by Tepperman et al. [5] dealt with the detection of sarcasm in speech, the area has received wide interest from the natural language processing (NLP) community. Not only speech, sarcasm detection has extended its sphere in various other data forms whether it's the tweets, movies or products reviews, and much more. Researchers have tried to evaluate and detect this form of a linguistic pattern via rule-based, supervised, semi-supervised and many other techniques. This continuous effort by the community people has resulted in new and interesting techniques for detection of sarcasm in textual data.
Some of the taxonomies already proposed in the literature are as:

1. In an article Camp [6] showed that usually there are four types of sarcasm: **Embedded:** This includes words and phrases that are having absurdity attached to them. **Like-prefixed:** These are the sentences and set of words which provide an implicit denial of the argument which has been made. **Propositional:** Here the sarcasm is present as a non-sentiment proposition but it has an inbuilt sentiment attached to it.
2. The famous **6-tuple representation,** proposed by Ivanke and Pexman [7] in 2003, was an important milestone in sarcasm identification in linguistics. They defined sarcasm consists of 6-tuples <S, H, C, u, p, p'>where:

   S= Speaker, H= Listener
   C= Content, u= Utterance
   p= Literal proposition
   p'= Intended proposition

   This can be read as *Speaker S generates an utterance u in Context C meaning proposition p but intending that hearer H understands p'.* They quoted a various example in the article itself to show that how the sentences can be divided or mapped on the 6 tuple representation.
3. Similarly, many studies have been proposed by Wilson [8] and Giora [10] where they mentioned about situational disparity theory and negation theory of sarcasm respectively.

Many researchers treat irony and sarcasm as strongly related and sometimes equate these two terms in their studies. In a study, Brown et.al.[10] gives warning that sarcasm is not a discrete logic or part of any linguistic phenomenon but the irony is part of the verbal expression. Hence, researchers take the liberty of using the term sarcasm through verbal irony is a better option. Considering all the definitions evolved by researchers in [9, 11, 12, 13, 14], we came to a conclusion that utterance of sarcastic words involves a shift in overall valence which can go either way: it could be a shift in the literally positive intended sentence to negative or the vice-versa. The occurrence of sarcasm in written and spoken interaction may work differently. In the spoken interactions, sarcasm is often marked with a modulation in the voice or special gesture which not used for normal conversation, this can also be named as an incongruent facial expression [15, 16]. As we do not have any special voice gesture or incongruent facial expression, thus, it is highly likely that most of the time researchers may not be able to pick up the utterance of sarcasm and interpret them, same has been said by authors in [17]. Some of the examples of sarcastic texts are:

1. *The best feeling in the world is definitely being ignored. I love it #keepdoingit #bestthingever #sarcasm.*
2. *Absolutely adore it when my bus is late #sarcasm.*

[3] `http://www.bbc.co.uk/worldservice/learningenglish/radio/specials/1210/ how to converse/page13.shtml`

3. *Wow GPRS data speeds are blazing fast :(. #beingsarcastic*
4. *Hes with his other woman: XBox 360. It's 4:30 fool. Sure I can sleep through the gun - fire #sarcastic #angry*

3.1. Techniques to detect sarcasm. Researchers from different corners of the world are trying to identify and work to solve the problem of detection of sarcasm in textual data. In an article, Davidov et.al. [18] wrote that recognition of sarcasm can benefit multiple communities working in various areas of natural language processing like a summarization of reviews, ranking systems, and dialog systems. In their technique, they performed a semi-supervised method on two different type of data sets i.e. one comprised of Amazon's product reviews and other has Tweets collected from Twitter. They had an accuracy of more than 75% on the both the datasets for identification of sarcasm and irony in them. They focused on the data specific features like punctuations, hashtags(#), sentence syntax and more.

Similarly, authors in [19] collected a training set of almost 78 thousand Dutch tweets. Their results implicate that irony and sarcasm in text is often signaled by a hyperbole, using intensifiers and exclamations, while the others are having an external marker. In another technique [20], authors presented an approach where a bootstrapping learning method to acquire a list of positive and negative tweets from sarcastic tweets have been discussed. Authors mentioned that the technique is still in its infantry and a lot of work has to be done. They have also discussed some ways in which future research can be done like sarcasm is often arisen from a description of a negative event.

One study [21] even got the precision of 98% while detecting sarcasm. They mentioned that it is of utmost importance especially for online social networks and product based services to identify and address the problem of sarcasm. They performed an analysis of the effect of irony and sarcasm on the overall polarity of the sentence/tweet. They designed a hashtag tokenizer for their solution and focused on the point that sarcasm found within hashtags can be detected more easily.

Some researchers have also tried to solve the problem by traditional ways of machine learning. In one such study [22], authors tried supervised learning technique of classification. They trained support vector machine (SVM) classifier with Bag-of-words as feature and TF-IDF as frequency measure. Though it's one of its kind studies the result proposed by authors suggest that only the Bag-of-words measure is not sufficient to claim the presence of irony and sarcasm in textual data. Many have worked in the direction of identification of sarcasm but still, much more is waiting to unfold.

Fig. 3.1 shows the tag cloud of the various features related to twitter or we can say twitter specific in nature that can be used for automatic sarcasm detection on twitter.

3.2. Why Twitter?. Twitter is an online social network and micro-blogging site, which enable users to send and read the short 140-character text messages known as tweets. *"The registered users can post and read tweets, while the unregistered users can only read the post. If a person likes the post of one user then he/she can share that particular tweet from the own profile and the process is known as a retweet. Twitter users have developed metadata annotation schemes which, shows that it compresses substantial amount of information into a comparatively tiny space".*

Currently, the user base of Twitter is over 650 million [23] worldwide, with almost 0.13 million new users joining daily. Over a billion, new tweets are being posted every month on this from a wide range of interest groups. *"Twitter is in the list of the most visited sites according to Alexa ranking [24], and has been described as "SMS of the Internet". Its large scale and streaming nature makes Twitter an ideal platform for monitoring events in real time".*

Recently, group of researchers from Carnegie Mellon University (CMU) came up with a study where they tried to detect sarcasm on contextualized basis [25]. They mentioned that most of the approaches in the past were treating sarcasm as a purely linguistic phenomenon, however by adding some extra-linguistic information such as the audience, the author and immediate communicative environment the accuracy of the protocol and further be increased. They also suggested that mere considering #sarcasm hashtag is not a natural indicator of sarcasm expressed in conversation, rather serves as an important communicative function to show the intent of the author to the audience, who may not otherwise be able to draw the correct idea or motive about the message. This article [25] created waves around the globe and got a lot of media attention too [26,27]. Fig 3.2 shows the basic structure of a tweet posted on Twitter.

FIG. 3.1. *Tag Cloud of Twitter features to detect Sarcasm*

FIG. 3.2. *Basic structure of a tweet*

4. Convolutional Neural Networks (CNN). Over past few years, machine learning techniques that are being applied to the artificial neural network have gained a lot of attention, especially they have played an important role in the designing of prediction models. Convolutional neural networks (CNNs or ConvNet) [28] are the type of feed forward artificial neural networks in which the pattern of connectivity between the neurons is inspired by the organization of animal visual cortex. Each neuron responds to stimuli in a restricted region of space called receptive field. Further, each field overlaps the next so that to tile the visual field. The response from each neuron within its receptive field can be approximated using the convolutional operation.

4.1. Convolutional Layer. The architecture of convolutional is formed by a stack of various independent layers that convert the input volume into output volume through a differentiated function. The CNN model helps capture spatial features of the given input which is in the form of a matrix. The model is less complex than a simple multilayer perceptron architecture. The simplicity of the model lies in the shared weights architecture design i.e. $N_W << N_x$ where N_W represents the number of weights and N_x represents the number of inputs. The weights are arranged as a window with shape $m \times n$ where $'m'$ and $'n'$ are hyper parameters defined by the user. These $m \times n$ weight matrices are called filters that run on the input matrix to get another matrix (C_1) with rows $x_r - m + 1$ and columns $x_c - n + 1$. The new matrix now goes under a **pooling** operation (explained in next section) which selects the maximum activated neuron from the matrix of the shape defined by the user. These new windows move in strides over the matrix C_1 and forms another matrix say P_1. The C_1 and P_1 combined forms one convolutional layer. The matrix P_1 is then passed to a fully connected layer that

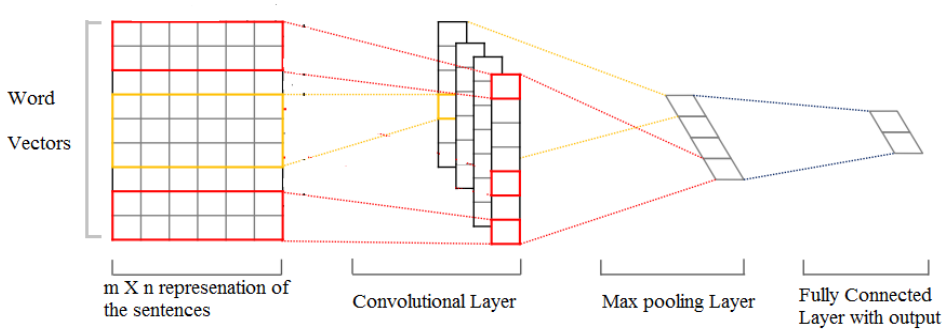


FIG. 4.1. *Architecture of a Single Layer Convolutional Neural Network*

TABLE 5.1
Total Number of Tweets collected from Twitter

Type of Tweet	Number of Tweets
Sarcastic	151896
Non-Sarcastic	330692
Total	482588

flattens the matrix and then classification can be performed by the simple multilayer perceptron model.

Fig 4.1 shows the various layer that comes into action when we perform the convolution function on our dataset. The final result is a fully connected layer having dropouts and softmax outputs.

4.2. Pooling Layer. In addition to convolutional layers, convolutional neural networks also contain *pooling layer*. They are usually used immediately after convolutional layers, as they play an important roll in simplifying the information in the output from the convolutional layer.

In detail, a pooling layer takes each feature map output from the convolutional layer and prepares a condensed feature map. For example, each unit in the pooling layer condenses a region of the previous layer. The pooling operation can be performed using techniques like *max-pooling, L2-pooling*.

5. Methodology. As discussed in section 3, Twitter can act as a perfect platform for our methodology. We started with the data collection, on to train our machine and finally testing our data against the algorithm we are using.

5.1. Data Collection. The advantage of using Twitter for our data set is, we can take as many samples as we want. Every day, people write new tweets which are sarcastic in nature. So, we started collecting the data using a Twitter API (Application Programming Interface) to stream the data. The API we used is tweeps [29]. In here we need to give keywords for collection of tweets. We carefully studied the usage of various keywords like *#Sarcasm, #Sarcastic, #beingsarcastic*. All the tweets related to these keywords were fetched and stored in the dataset for analysis purpose. The requirement to create our own data set was the unavailability of data set related to sarcasm in tweets.

As we are using classification technique which is part of supervised learning, hence we need to have training as well as a test set to check the accuracy and efficiency of the algorithm. We collected almost 0.5 million tweets which included both sarcastic and non-sarcastic tweets. The details of collected tweets are mentioned in Table 5.1.

5.1.1. Drawback of Twitter. The major disadvantage of taking Twitter into consideration is that the stream of data we receive is *too noisy*. By the term noisy, we mean to say that lot of retweets, social interaction, comments all are part of the data stream. Some people might use the hashtag *#sarcasm* to point out that they are trying to be sarcastic, but it is not sarcastic to others or in general. In another case, people may be

clearly sarcastic but they have not used *#sarcasm* hashtag. One such case can be when one person's tweet is a response to another and which is sarcastic to the context but not otherwise. All such problem adds to the noise we encounter when we stream data from the twitter.

5.2. Preprocessing of Data. Once we have the dataset, its time to clean it up to remove the noise we discussed in the previous subsection. For starters,

1. We removed every tweet in which the sarcasm is either present in the link or they are a response to another tweet.
2. All the tweets starting with `http://` are discarded along with those starting with `@`.
3. All the tweets which were not in English were again not considered for the analysis purpose. To maximize the collection of English tweets we even considered location based tweets i.e. a tweet coming from New York or London is likely to be in English as compared to one coming from Paris.
4. We even removed tweets which are too small for analysis purpose i.e. at least 3 or more words should present.
5. All the retweets are removed. Retweets are the tweets which have been forwarded by a twitter user but are not generated by him/her.
6. We lastly removed all the hashtags, friends mentions, and any non-ASCII value from the tweet.

5.3. Word to Vector Conversion. Deep neural network architectures are giving ground breaking results in every field of artificial intelligence and, Natural Language Processing is one of it. The automation of this field is much needed since the application of basic NLP techniques is tedious and a time-consuming task. The idea is to model the words into an appropriate vector space such that, we can use those vectors to perform any task. Many researchers gave their model to capture appropriate word vectors for better learning. Most famous ones are CBOW [30], skip-gram word2vec [31] and GloVe [32].

We are using the skip-gram word2vec model to convert sentences into vectors that will be further passed and will help us find out/ predicting the surrounding words in the sentence or a document. Formally, given a sequence of words which are being used for the training purpose say W_1, W_2, W_3, ..., W_T, the objective of this technique is to maximize the average log probability given by equation (5.1):

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} logp(w_{t+j} \mid w_t) \tag{5.1}$$

In this equation, c is the size of the training context, which is a function of center word wt. Increasing the size of the c can result into more training examples thus leading us to a higher accuracy but at an expense of more training time. The basic skip-gram formulation defines $p(w_{t+j} \mid w_t)$ using a softmax function given by equation (5.2):

$$p(w_O \mid w_T) = \frac{\exp(v'^T_{wO} v_{wI})}{\sum_{w=1}^{W} \exp(v'^T_w v_{wI})} \tag{5.2}$$

where vw and $v'w$ are the input and output vectors of w and W is the total words in the vocabulary. This formulation is often impractical due to the very large computational cost of computation even for a small δ value.

5.4. Modeling sentences using CNN. Once we get appropriate word vectors from the skip-gram word2vec model, then we take each sentence at a time and change the word in it to a vector. The given sentence will now be a matrix where each row is a word vector. As mentioned, the CNN takes matrices as input. These sentences are padded with an unknown token <UNK> till the size is equal to the maximum length sentence(say S_nn) in the training data. Vector for <UNK> is 0 in all dimensions. The matrix of a sentence has S_n rows and W_d column where W_d are the dimensions of the word vectors. We have taken 150 filters of shape $(i \times W_d)$ where i = [3,4,5]. Each of them stride over the input matrix and results in a matrix C_1 of shape $(S_n \times i)$ for each filter. Since the matrices are of different shapes, we perform max-pooling on each matrix and get a feature vector whose size is equal to the number of filters i.e $(150 \times i)$ in our case. The feature vector is now connected to two neurons that will help in classification.

TABLE 6.1
Accuracy percentage variation with number of iterations

Number of Iterations	Accuracy (%)
1000	83.5
2000	85.3
3000	86.9
4000	87.5
5000	88.2
6000	88.9
7000	89.3
8000	89.7
9000	89.9
10000	89.9

5.5. Classification. In the final step, we parse the feature vector to the two neuron units for the classification. The labels of data are [0,1] and [1,0] for sarcastic and non-sarcastic tweets respectively. The two neuron units classify the data on the basis of the highest activation received from the product of feature vector and respective weights. For the calculation of the optimized function, we used *Adam* [33], an algorithm for first-order gradient-based optimization of stochastic objective functions. The idea behind the usage of this objective function is it's very straight forward to implement, computationally very efficient, needs less memory and is very well suitable for the data sets having a lot of noise or very sparse in nature. Also, the hyperparameters have intuitive interpretations and typically require little tuning. Once passed, the system keeps on evaluating the data set with more number of iterations every time. For our experiment, we have taken readings from 1000 to 10000 iterations with a space span of 1000 between each reading.

6. Results. We have trained our model on a large dataset of tweets, where a tweet is automatically labeled as sarcastic or not, depending on the occurrence of words and learning from the training set. Our experimental setup has more than 0.45 million tweets in it helping to improve the accuracy of the system. We are considering a batch size of 128 tweets at a time to train the model and then proceed similarly for the next batch.

Our system has achieved an accuracy of 89.9% after running the convolution for 10000 iterations as shown in Table 6.1. With every 1000 iterations, we see a significant improvement in the accuracy till 9000 iterations, After this, the result changes with a very minimal value. The major reason for this decline in the improvement of accuracy is the amount of randomness in the data reduces thus the accuracy becomes stagnant. The model uses only a single layer of convolution to achieve accuracy that is at par or even better than many models proposed in the literature. The graph in Fig 6.1. shows the comparison between the number of iterations and percentage accuracy.

7. Conclusion and Discussion. We have discussed an approach in this article that uses convolution neural networks for the classification and detection of sarcasm in textual data taken from a microblogging site known as Twitter. The study reveals that the algorithm performs well with single layer convolution and even out perform many existing techniques. Also, the detection of sarcasm is an open problem as many new dimensions are still to be explored. It is inherited and has become part of our daily routine conversation i.e. verbal or textual.

Most of the researchers mentioned about the amalgamation of scientific theories with practical engineering goals to analyze sarcasm in natural language text. This will not only lead to better understanding of natural language opinions and one can easily reduce the gap between unstructured information and structure data. An accurate measure to detect sarcasm will not only help researcher but it will also benefit the businesses and product services. In future, we are planning to further evaluate our technique and try to in-cooperate the multi-layer or dynamic layer convolution into consideration.

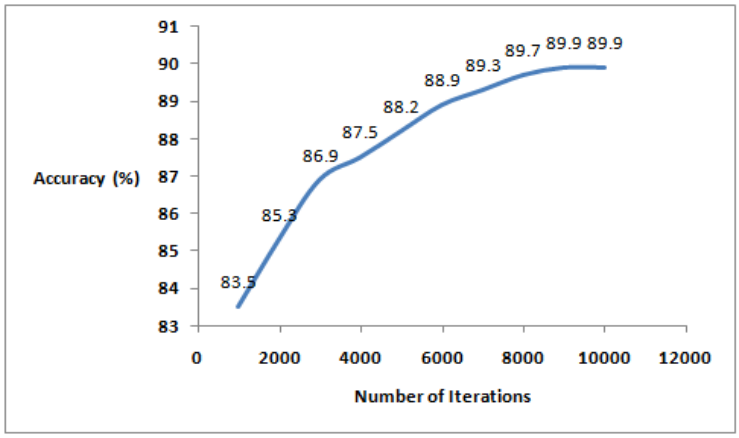


FIG. 6.1. *Accuracy vs. Number of Iterations*

REFERENCES

[1] HORRIGAN, JOHN A. *Online shopping. Pew Internet & American Life Project Report* 36 (2008).
[2] TUMASJAN, A., SPRENGER, T. O., SANDNER, P. G., AND WELPE, I. M. *Predicting elections with twitter: What 140 characters reveal about political sentiment.* ICWSM, 10, 178-185.
[3] MEHNDIRATTA, P., SACHDEVA, S., SACHDEVA, P., AND SEHGAL, Y.. *Elections Again, Twitter May Help!!! A Large Scale Study for Predicting Election Results Using Twitter.* In Big Data Analytics (pp. 133-144). Springer International Publishing.
[4] CHUNG, J. E., AND MUSTAFARAJ, E. *Can collective sentiment expressed on twitter predict political elections?.* In AAAI (Vol. 11, pp. 1770-1771)
[5] TEPPERMAN, J., TRAUM, T.R., AND NARAYANAN, S. *" yeah right": sarcasm recognition for spoken dialogue systems..* In INTERSPEECH. Citeseer
[6] CAMP, E. *Sarcasm, Pretense, and The Semantics/Pragmatics Distinction.* 46, 4, 587634, (2012).
[7] IVANKO, S.L. AND PEXMAN, P.M., *Context incongruity and irony processing.* Discourse Processes 35, 3, 241279, (2003).
[8] WILSON, D., *The pragmatics of verbal irony: Echo or pretence?* Lingua 116, 10, 17221743, (2006).
[9] GIORA, R., *On irony and negation. Discourse processes* 19, 2, 239264, (1995).
[10] BROWN, R. L. *The pragmatics of verbal irony. Language use and the uses of language*, 111-12, (1980).
[11] HAMAMOTO, H.*Irony from a cognitive perspective.* Pragmatics and beyond new series, 257-270, (1998).
[12] ATTARDO, S. *Irony as relevant inappropriateness.* Journal of pragmatics, 32(6), 793-826, (2000).
[13] PARTINGTON, A.*Irony and reversal of evaluation.* Journal of pragmatics, 39(9), 1547-1569, (2007).
[14] SPERBER, D., AND WILSON, D. *Relevance: Communication and cognition* (Vol. 142). Cambridge, MA: Harvard University Press, (1986).
[15] BRYANT, G. A., AND TREE, J. E. F. *Is there an ironic tone of voice?.* Language and speech, 48(3), 257-277, (2005).
[16] JAHANDARIE, K. *Spoken and written discourse: A multidisciplinary perspective.* Greenwood Publishing Group, (1999).
[17] BURGERS, C. F.*Verbal irony: Use and effects in written discourse*, Ipskamp, Nijmegen, The Netherlands, (2010).
[18] DAVIDOV, D., TSUR, O., AND RAPPOPORT, A.*Semi-supervised recognition of sarcastic sentences in twitter and amazon.* In Proceedings of the Fourteenth Conference on Computational Natural Language Learning (pp. 107-116). Association for Computational Linguistics (2010).
[19] LIEBRECHT, C., KUNNEMAN, F., AND VAN DEN BOSCH, A.*The perfect solution for detecting sarcasm in tweets #not.* WASSA 2013, 29, (2013).
[20] RILOFF, E., QADIR, A., SURVE, P., DE SILVA, L., GILBERT, N., AND HUANG, R. *Sarcasm as Contrast between a Positive Sentiment and Negative Situation.* In EMNLP (pp. 704-714) (2013).
[21] MAYNARD, D., AND GREENWOOD, M. A. *Who cares about Sarcastic Tweets? Investigating the Impact of Sarcasm on Sentiment Analysis.* In LREC (pp. 4238-4243), (2014).
[22] DAVE, A. D., AND DESAI, N. P. *A Comprehensive Study of Classification Techniques for Sarcasm Detection on Textual Data.*
[23] TWITTER STATISTICS, `www.statisticsbrain.com/twitter-statistics/`
[24] ALEXA: THE WEB INFORMATION COMPANY, `www.alexa.com`
[25] BAMMAN, D., AND SMITH, N. A. *Contextualized sarcasm detection on Twitter.* In Proceedings of the 9th International Conference on Web and Social Media (pp. 574-77). Menlo Park, CA: AAAI (2015).
[26] *Researchers are teaching computers sarcasm with Twitter* (2016): `http://thenextweb.com/twitter/2016/01/22/researchers-are-teaching-computers-sarcasm-with-twitter/#gref`
[27] KNIGHT, W. (2016, January) *My Favorite Thing about the Internet? Definitely the Sarcasm:* `https://www.technologyreview.com/s/545936/my-favorite-thing-about-the-internet-definitely-the-sarcasm/`
[28] LECUN, Y., BOTTOU, L., BENGIO, Y. AND HAFFNER, P.*Gradient-based learning applied to document recognition.* Proceedings

of the IEEE, 86(11), pp.2278-2324, (1998).
[29] Researchers are teaching computers sarcasm with Twitter (2016): http://thenextweb.com/twitter/2016/01/22/researchers-are-teaching-computers-sarcasm-with-twitter/#gref
[30] Mikolov, T., Chen, K., Corrado, G. and Dean, J. *Efficient estimation of word representations in vector space.* arXiv preprint arXiv:1301.3781, (2013).
[31] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J.*Distributed representations of words and phrases and their compositionality.* In Advances in neural information processing systems (pp. 3111-3119), (2013).
[32] Pennington, J., Socher, R. and Manning, C.D. *Glove: Global Vectors for Word Representation.* In EMNLP (Vol. 14, pp. 1532-1543), (2014).
[33] Kingma, D. and Ba, J. *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980, (2014).

Edited by: Kavita Sharma
Received: May 5, 2017
Accepted: Sep 2, 2017

DOI 10.12694/scpe.v18i3.1303
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 18, Number 3, pp. 229–241. http://www.scpe.org

DESIGN AND ANALYSIS OF MODIFIED SIQRS MODEL FOR PERFORMANCE STUDY OF WIRELESS SENSOR NETWORK

RUDRA PRATAP OJHA, GOUTAM SANYAL *, PRAMOD KUMAR SRIVASTAVA†, AND KAVITA SHARMA ‡

Abstract. The dynamics of worm propagation in the Wireless Sensor Networks (WSNs) is one of the fundamental challenge due to critical operational constraints. In this paper we propose a modified Susceptible-Infectious-Quarantined-Recovered-Susceptible (SIQRS) model based on epidemic theory. The proposed model demonstrate the effect of quarantined state on worms propagation in WSNs. This model incorporates communication radius, area of communication and the associated node density. The spreading dynamics of worms defined with the help of Basic Reproduction Number (R_0) and if R_0 is less than or equal to one the worm-free equilibrium is globally asymptotically stable, and if R_0 is greater than one the worm will persists in the system. This model formulated by differential equations and explain the process of worm propagation in WSNs. We also study the effect of different parameters on the performance of system. Finally, the control mechanism and performance of the proposed model is validated through extensive simulation results.

Key words: Epidemic model, Basic reproduction number, Stability, Wireless Sensor Network, Communication radius, Node density

AMS subject classifications. 68M10, 90B18

1. Introduction.

Data communication is one of the primary requirements of modern society. There are different ways of data communication,it may be wired and wireless. The wireless sensor network is also a kind of wireless communication and it having great potential of application like military, patient health monitoring, vehicle traffic monitoring, battlefield, environmental monitoring etc.[1]. Wireless sensor network is a collection of large number of sensor nodes and it is a small device equipped with memory, processing unit, energy source and communication units. The Sensor nodes can be deployed in any type of terrain for collection of data from surroundings and send them to sink node via neighbor nodes. There are some limitations of sensor nodes for example communication range, energy, memory etc. therefore data delivered at the sink node in multi-hop [2]. The Sensor nodes are scattered in a hostile environment with restricted power and charging is very difficult task at deployed location. In inaccessible region, the sensors nodes are deployed by robot or airplane. Due to large applications of WSNs it becomes one of the hot topic for researchers. Energy consumption and increasing the lifetime of WSNs methods has been developed[2, 3, 4], another method related to network topology [5] and placement[6] of nodes has been also studied. WSNs have resource constraints therefore it has a weak defense and soft target for malware attack on nodes [7]. The sensor nodes can be easily targeted by software attacks like virus or worm. Nowadays, wireless devices are targeted by malicious codes very easily and spread from device to device through wireless communication for example Bluetooth and Wi-Fi[8, 9]. The controlling of worm propagation is one of the difficult task. Mathematical modeling is an important tool for analyses the dynamics of worm propagation in wireless sensor network. A worm attack on a certain node in wireless sensor network and its gets infected, then infection spread through the neighbour nodes in the entire network [10]. Therefore, security mechanism against worms attack for wireless sensor network has essential practical importance. Since, there is fundamental similarity between biological worm transmission based on epidemic model and the software generated worm in wireless network.

The social researchers used the concept of epidemic model [11, 12, 13] comprehensively and could be correctly applied to worms spread in WSN. There are some allied applications of epidemic models in the literature of wireless network [8, 13, 14, 15, 16]. The spreading of virus over the internet has been broadly studied by researchers based on the concept of epidemic model [13, 14, 17, 18, 19] and some epidemiological model extensively developed for WSN [8, 10, 16]. A simple algorithm for information diffusion SI model [14] was proposed for analysis of worm propagation in mobile ad hoc networks and developed an expression that describe the relation between rate of infection and node density. Another modified SI model [10] was proposed

*National Institute of Technology,Durgapur(rpojha@gmail.com,nitgsanyal@gmail.com).
†Galgotias College of Engineering Technology(pramodpooja59@gmail.com).
‡National Institute of Technology, Kurukshetra(kavitasharma_06@yahoo.co.in)

to improve the antivirus capability of networks by leveraging sleep mode and capture both spatial and temporal dynamics simultaneously. This model was verified with the help of extensive simulation.

In this paper, we study the temporal and spatial dynamics and possible attacking behavior of worms in the wireless sensor network. The proposed model depicts worm propagation behavior in wireless sensor network with quarantine state in real world. The meaning of quarantine is to forcefully isolate and stop working of nodes. When infected nodes are detected at any time in the network then they promptly be isolated by detection program. We can stop the spreading of worm and improve the security of network as well as increase the lifetime of wireless sensor network by using this model. Remaining part of the paper is ordered as follows.Section 2 include the related work; Section 3 describes the proposed model. Modeling and analysis of proposed model is given in section 4. Equilibrium and stability studied in section 5. Simulation results and performance analysis in section 6. Section 7 conclusion and future scope.

2. Related work. Studies the process of worm propagation on the Internet based on epidemic theory was proposed by [19]. There are some worm propagation conventional model on the internet:SIS,SIR,Two-factor and IWM model [20, 21]. These models comprise of differential equation,and they can successfully portray the qualities of worm spread on the internet. Practically speaking, the SIR model is an expansion of SIS model and it is generally connected in investigating the flow of worm proliferation on the internet. In any case, the above models are not especially intended for WSNs. Therefore, these models do not bolster the worm propagation with the vitality utilization of nodes in WSNs. There are different epidemic models have been examined by various researchers that investigate the dynamic behavior of worm spreading and controlling in WSNs.

In [22] the author study the virus spreading behavior of SIRS model in the wireless sensor network that includes the degree of nodes, topological connection and develops the method for the prevention and controlling of local infection.

Pietro et al.[23] proposed an epidemic model for information survivability in unattended WSNs to ensure the quality of service and node energy in presence of attacker.

In [24] presented a SI model with inclusion of communication radius and node density that illustrate the effect of these parameters on spreading of virus in wireless sensor network with MAC mechanism. But this model did not include the recovery of nodes and other method of protection due to attack of virus, overcome this problem in [25] introduced a model that was SIR-M in which one more state recovery included, and carry out maintenance task in the sleep state of node and recover the infected node without any overhead and enhance the lifetime of WSNs and model was demonstrated by the simulation.

Author [26] proposed an improved SIRS model with inclusion of dead node that describe the process of worm propagation and also included the energy consumption of nodes in WSNs. But this model did not include a number of significant characteristics of model for example reproduction number, stability condition, equilibrium points and delay of network. These missing uniqueness are discussed by some different authors. In [27] author calculate the reproduction number and examines its effect on the wireless sensor network as well as also found the condition of worm free equilibrium and local stability condition of the proposed model. This model was verified by simulation with the help of MATLAB. Zhu and Zhao [28] have discussed another model with inclusion of dead node to study the behavior of malware propagation in wireless sensor network. Numerical proof and simulation shows the dynamic behavior of worm propagation in wireless sensor network but this model do not focus on characteristics and specifications of particular sensor node. These issues are talked by [29],proposed an individual-based models that determine specific attributes of each component of model and beat the deficiency of the past model by the consideration of each device characteristics and evaluate each individually. Ojha et al.[30] proposed a model to study the effect of undetected infectious nodes and effect of treatment. They also, calculate the threshold value and analyze its effect on the wireless sensor network and discuss the local and global equilibrium stability of worm free state. These are proved with the help of simulation.

In [31] a control mechanism developed to efficiently hold down the worm outbreak and escape the network breakdown. This model includes number of nodes, transmission range of sensors,node density etc. and verified by simulation. Keshri and Mishra [32] proposed an epidemic model with inclusion of the two delay period and study its effect on different class of the model. They also found the local and global stability of worm free equilibrium point of the system and calculate its reproduction number of the proposed system as well as observed its effect on wireless sensor network, communication radius,node density and quarantined class have not

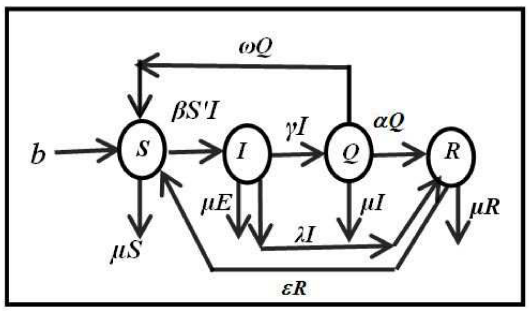


FIG. 3.1. *Transition relationship of node states*

TABLE 3.1
Parameters Used

S.No	Symbol	Meaning of Symbol
1	S	Susceptible
2	I	Infected
3	Q	Quarantined
4	R	Recovered
5	b	Birth rate
6	μ	Death rate
7	β	Rate of infection
8	α	Probability of moving from exposed node to infectious category
9	γ	Recovery rate of an infectious node
10	ω	Probability of moving from Quarantined node to Susceptible node
11	ε	probability at which a recovered node become a susceptible node

been considered in proposed model. A SIRS model was proposed by [33] in which consider the communication radius and node density as well as calculate the reproductive number and found the Eigen values for stability verification of the system. But this model have not efficient mechanism to control the quick spreading of worm in the network. We extend the proposed model with introduction of quarantined class. The quarantined state plays a significant role and sends the infectious nodes into isolation mode and removes the worm from network without any overhead and stretch the life span of network. This model overcomes the insufficiency of existing models.

3. The proposed model. We consider the total numbers of nodes in the network are $N(t)$ at any time t which is uniformly distributed in the region $L \times L$ with average density ρ and radio range of sensor node is r. There are four states in the proposed model-Susceptible state S The nodes which are not infected but vulnerable to the worm.Infected state I -The infected node which is capable to infecting other nodes. The Quarantined Q -The isolation state of node and Recovered state R -The recovered nodes become immune.In the proposed model, we have assumed that the deployment of sensor nodes are uniform and random in two dimensional areas. Each sensor nodes having a radio range r and coverage area of each node is $\pi \times r^2$. The significance of coverage area is to at least one sensor node lies within the sensing range because data delivered to the sink through these nodes.Each node detects the events and reported to the sink. Initially, consider each and every node to be susceptible and can be attack by worm. The relationship between the transitions states is depicted by Fig. 3.1. There is some limitation of the model because all sensor nodes are homogeneous.

4. Modeling and Analysis. We assume that the communication area of a sensor node is πr^2 and the density of a susceptible node in a unit area is $\rho(t) = S(t)/(L \times L)$. The total number of neighboring nodes which lie in the communication range of a sensor node is given as : $S^{'}(t) = \rho(t)\pi r^2 = \frac{S(t)\pi r^2}{L \times L}$. According to Fig 3.1, we consider following mathematical model of worm propagation.

$$\left.\begin{aligned} \frac{dS}{dt} &= b - \frac{\pi r^2}{L^2}\beta SI + \omega Q + \varepsilon R - \mu S \\ \frac{dI}{dt} &= \frac{\pi r^2}{L^2}\beta SI - (\mu + \lambda + \gamma)I \\ \frac{dQ}{dt} &= \gamma I - (\alpha + \omega + \mu)Q \\ \frac{dR}{dt} &= \alpha Q + \lambda I - (\mu + \varepsilon)R \end{aligned}\right\} \tag{4.1}$$

Let $\zeta = \frac{\pi r^2}{L^2}\beta$. Then the above system (4.1) can be written as:

$$\left.\begin{aligned} \frac{dS}{dt} &= b - \zeta SI + \omega Q + \varepsilon R - \mu S \\ \frac{dI}{dt} &= \zeta SI - (\mu + \lambda + \gamma)I \\ \frac{dQ}{dt} &= \gamma I - (\alpha + \omega + \mu)Q \\ \frac{dR}{dt} &= \alpha Q + \lambda I - (\mu + \varepsilon)R \end{aligned}\right\} \tag{4.2}$$

5. Existence of positive equilibrium. For equilibrium points we have

$$\frac{dS}{dt} = 0, \frac{dI}{dt} = 0, \frac{dQ}{dt} = 0, \frac{dR}{dt} = 0$$

and after a straight forward calculation, we get equilibrium points as $P_0 = (S_0, I_0, Q_0, R_0) = (\frac{b}{N}, 0, 0, 0)$ for worm free state and $P^* = (S^*, I^*, Q^*, R^*)$ for endemic state, where

$$S^* = \frac{\lambda + \gamma + \mu}{\zeta} I^* = (1 - \frac{1}{R_0})\frac{b(\varepsilon + \mu)(\mu + \alpha + \omega)}{A}, Q^* = (1 - \frac{1}{R_0})\frac{\gamma(\varepsilon + \mu)}{A}, R^* = (1 - \frac{1}{R_0})\frac{\alpha(\gamma + \lambda) + \lambda(\mu + \omega)}{A}$$

where $A = (\mu + \alpha)(\lambda + \mu) + (\mu + \varepsilon) + (\gamma + \varepsilon + \omega) + (\alpha\gamma + \omega\lambda)$ and R_0 [34] is the basic reproduction number and given by $R_0 = \frac{b\zeta}{\mu(\mu + \lambda + \gamma)}$. It is clear that P^* exist and unique if and only if $R_0 > 1$.

5.1. Stability analysis of the worm free equilibrium.

THEOREM 5.1. *The worm free equilibrium (W F E) is locally asymptotically stable if $R_0 < 1$ otherwise unstable for $R_0 > 1$.*

Proof. In order to check the stability of point ,we will find all the Eigenvalue values of the variational matrix is given by

$$J(P_0) = \begin{bmatrix} -\zeta I_0 - \mu & \zeta S_0 & \omega & -\varepsilon \\ \zeta I_0 & \zeta S_0 - (\lambda + \gamma + \mu) & 0 & 0 \\ 0 & \gamma & -(\alpha + \mu + \omega) & 0 \\ 0 & 0 & \alpha & -(\varepsilon + \mu) \end{bmatrix} \tag{5.1}$$

□

Eigen values of (5.1) are: $\theta_1 = -\mu$, $\theta_2 = -(\mu + \alpha + \omega)$, $\theta_3 = (\varepsilon + \mu)$, $\theta_4 = (R_0 - 1)(\frac{1}{\gamma + \lambda + \mu})$.

It is clear that all eigen values of the variational matrix are negative when $R_0 < 1$. Therefore,the system is locally asymptotically stable at worm free equilibrium.Furthermore, the following theorem holds.

THEOREM 5.2. *The worm free equilibrium is said to be globally asymptotically stable if* $R_0 \leq 1$.

Proof. Consider the Lyapunov function $L(t)$: $R^4 \longrightarrow R^+$ defined by $L(t) = \eta I(t)$,now taking time derivative of $L(t)$ with respect to time t is $\frac{dL(t)}{dt} = \eta_1 I(t) \Rightarrow \frac{dL(t)}{dt} = \eta(\zeta SI) - (\gamma + \mu + \lambda)I) = \eta(\zeta S - (\gamma + \mu + \lambda))I$. After suitable assumption of $\eta = \frac{1}{(\gamma + \lambda + \mu)}$ we get $\frac{dL}{dt} = [\zeta S - (\gamma + \lambda + \mu] = [\frac{\zeta b}{(\mu + \gamma + \lambda)\mu} - 1]I = (R_0 - 1)I$. It is clear that $\frac{dL}{dt} = 0$ only when $I = 0$. Therefore,the maximum invariant set $\Gamma = (S, I, Q, R) \in R_4^+$ is the singleton set. Therefore, the global stability of worm free equilibrium P_0 when $R_0 \leq 1$ follows from Lasalle invariance principle [35]. □

5.2. Stability analysis of Endemic equilibrium.

THEOREM 5.3. *Worm endemic state(WES)is locally asymptotically stable if* $R_0 > 1$.

Proof. In order to check the stability at point P^* we will find all the Eigen values of the variational matrix:

$$J(P^*) = \begin{bmatrix} -\zeta I^* - \mu & \zeta S^* & \omega & \varepsilon \\ \zeta I^* & \zeta S^* - (\lambda + \gamma + \mu) & 0 & 0 \\ 0 & \gamma & -(\alpha + \mu + \gamma) & 0 \\ \omega & 0 & 0 & -(\varepsilon + \mu) \end{bmatrix} \tag{5.2}$$

Therefore, the corresponding characteristic equation for the above matrix is given by

$$\nu^4 + D_1\nu^3 + D_2\nu^2 + D_3\nu + D_4 = 0 \tag{5.3}$$

where

$D_1 = \zeta Q_1(R_0 - 1) + 3\mu + \alpha + \omega + \varepsilon$

$D_2 = (\mu + \alpha + \omega)(\mu + \varepsilon) + (\zeta Q_1((R_0 - 1) + \mu)(\alpha + 2\mu + \omega + \varepsilon) + Q_2$

$D_3 = \zeta Q_1(R_0 - 1)(\omega\gamma + \varepsilon\lambda) + (\mu + \alpha + \omega)(\mu + \varepsilon)(Q_2 - 1)$

$D_4 = \zeta Q_1(R_0 - 1)(\gamma\varepsilon\alpha + \omega\gamma(\mu + \varepsilon) + Q_3)$

where

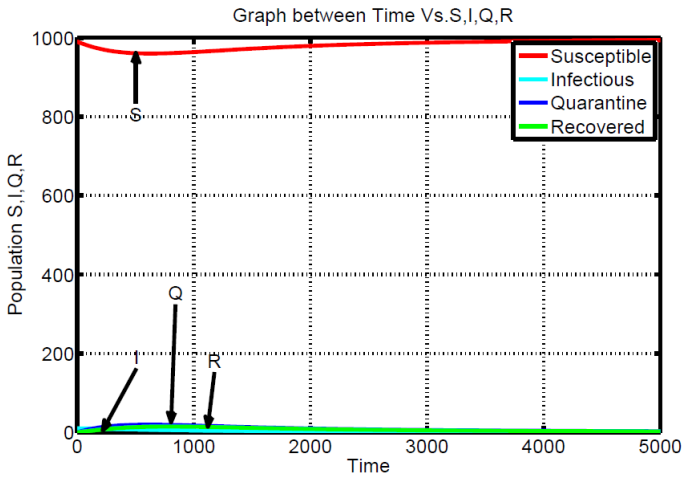
$Q_1 = \frac{b(\mu + \alpha + \omega)(\mu + \varepsilon)}{AR_0}$,

$Q_2 = -\zeta^2 I^* S^*$,

$Q_3 = -(\alpha + \mu + \omega)[(\mu + \gamma)(\mu + \varepsilon) + \mu\lambda]$

It is clear that all D_1, D_2, D_3, D_4 are positive when $R_0 > 1$,by a direct calculation $G_1 = D_1 > 0$,$G_2 = D_1D_2 - D_3 > 0$,$G_3 = D_3G_2 - D_1^2D_4 > 0$ and $G_4 = D_4, G_3 > 0$. Therefore, according to Routh Hurwitz criteria, all the roots of equation (5.3) have negative real parts. Therefore,the endemic equilibrium P^* is locally asymptotically stable when $R_0 > 1$. This completes the proof. □

6. Simulation and Performance analysis. It has been shown in the above analysis that R_0 is a threshold value. When R_0 less than equal to one,the fraction of infected nodes vanishes and when $R_0 > 1$,the fraction of infected nodes persist in the system. We analyzed the behavior of worm propagation to study the effect of parameters on performance. The proposed model has been simulated with the help of MATLAB and obtained results analyzed below.



$N = 1000; b = 1; L = 10; \beta = 0.0002; \mu = 0.001; \gamma = 0.002; \varepsilon = 0.0003; \omega = 0.0001; \alpha = 0.002; \lambda = 0.0005; r = 0.7$

FIG. 6.1. *Dynamic behavior of model when communication radius is 0.7*

6.1. Communication Radius of nodes r. As we have shown that

$$R_0 = \frac{b\zeta}{\mu(\mu+\gamma+\lambda)}, \text{ where } \zeta = \frac{\pi r^2 \beta}{L^2}.$$

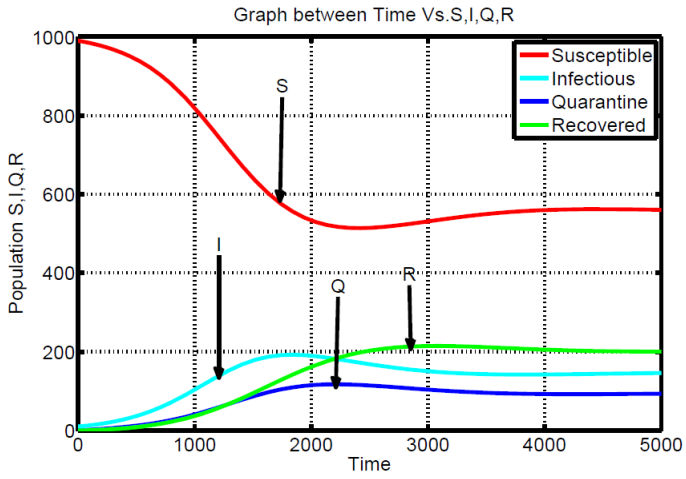
From this equation, we find threshold radius $r_c = \text{L}\sqrt{\frac{(\mu+\gamma+\lambda)\mu}{\pi\beta}}$ i.e when $r \leq r_c, R_0 \leq 1$ from theorem 5.2, it is clear that the worms in wireless sensor networks can be eradicated and system will stable at worm free equilibrium, when $r > r_c, R_0 > 1$ according to theorem 5.3, worms in wireless sensor networks will be present consistently and system will stabilize at the endemic equilibrium.

We take following values of parameters as $N = 1000; L = 10; b = 1; \beta = 0.0002; \mu = 0.001; \gamma = 0.002; \varepsilon = 0.0003; \omega = 0.0001; \alpha = 0.002; \lambda = 0.0005$. After calculation, we get $r_c = 0.746542$ the initial values of susceptible, exposed ,infected and recovered nodes in wireless sensor networks are $S(0) = 990, I(0) = 10, Q(0) = 0$ and $R(0) = 0$ and takes different values of r, simulation results are described in Fig.6.1, 6.2 and 6.3. When $r = 0.7 < r_c$, Fig.6.1 shows that system (4.2) stabilizes at worm free equilibrium and the simulation results are consistent with theorem 5.2.

When $r = (1.0 \text{ and } 1.4) > r_c$, Figs. 6.2 and 6.3 shows that system (4.2) stabilizes at the endemic equilibrium and the simulation results are consistent with theorem 5.3. Communication radius plays an important role for connectivity. Therefore, connectivity is a function of communication radius. From Figs. 6.1, 6.2 and 6.3 we see that when communication radius of nodes increases connectivity increases but the value of R_0 also increases. When the communication radius less than r_c the network will be stable and worm free. As the communication radius increases R_0 also increases and lead to the failure of the system.

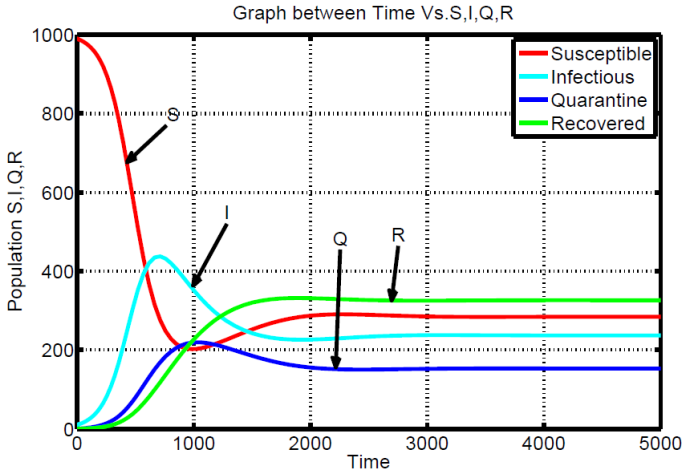
Fig. 6.4 shows that relation between infected and recovered nodes with variation in the parameters. We observed that when the value of α increases the more number of nodes are recovered smoothly. We also find that if we increase the value of γ, less number of nodes get infected and recovered in short time. This experiment shows the effect of quarantined state on sensor network.

Fig. 6.5 shows the response of the number of susceptible nodes with respect to change in the parameters. We observed that gradually the number of susceptible nodes decrees and recovered nodes increases.



$N = 1000; b = 1; L = 10; \beta = 0.0002; \mu = 0.001; \gamma = 0.002; \varepsilon = 0.0003; \omega = 0.0001; \alpha = 0.002; \lambda = 0.0005; r = 1.0$

FIG. 6.2. *Dynamic behavior of model when communication radius is 1.0*



$N = 1000; b = 1; L = 10; \beta = 0.0002; \mu = 0.001; \gamma = 0.002; \varepsilon = 0.0003; \omega = 0.0001; \alpha = 0.002; \lambda = 0.0005; r = 1.4$

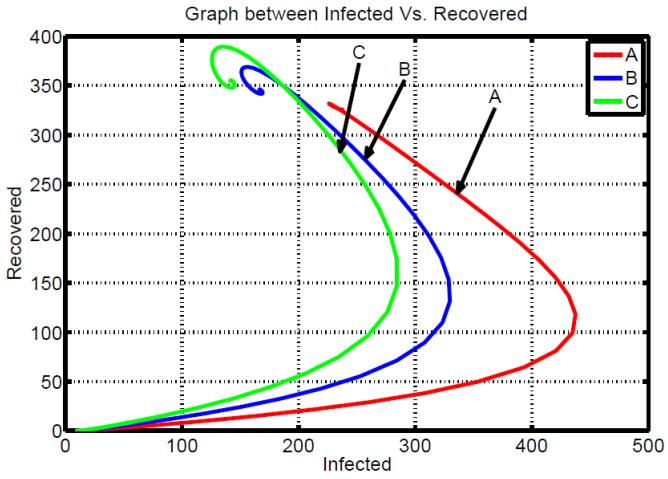
FIG. 6.3. *Dynamic behavior of model when communication radius is 1.4*

6.2. Node distributed density. The threshold value of node density ρ_{th} is given by

$$\rho_{th} = \frac{N(\mu + \gamma + \lambda)\mu}{b\pi r^2 \beta}$$

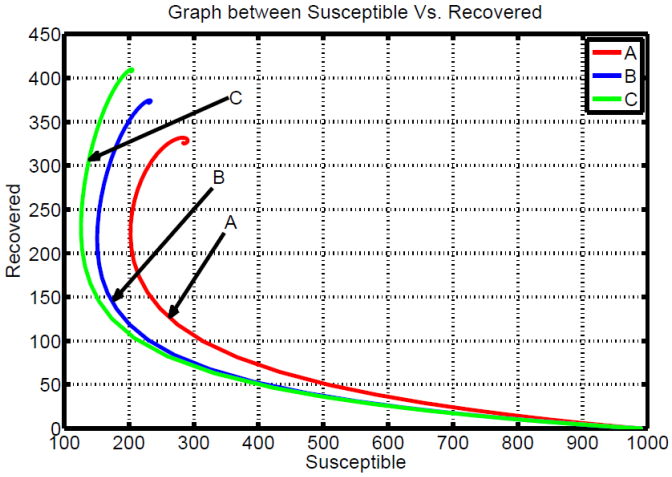
i.e., when $\rho \leq \rho_{th}, R_0 \leq 1$ then the system has only equilibrium and is globally asymptotically stable; when $\rho > \rho_{th}, R_0 > 1$,system has only one endemic equilibrium which is locally asymptotically stable.

We take following values of parameter- $N = 1000; b = 1; \beta = 0.0002; \mu = 0.001; \gamma = 0.002; \varepsilon = 0.0003; \omega = 0.0001, \alpha = 0.002; \lambda = 0.0005; r = 1$. After calculation ,we get $\rho_{th} = 5.573248$. Initial values of susceptible, exposed ,infected and recovered nodes in wireless sensor networks are $S(0) = 990, I(0) = 10, Q(0) = 0$ and $R(0) = 0$ and When $L = 11.18$ and 8.45 ,we can get $\rho = 8$ and 14 using these various values, simulation results are shown in Figs. 6.6 and 6.7. When $\rho =< \rho_{th}$, the system (4.2) stabilizes at worm free equilibrium and the simulation results are consistent with the theoretical analysis. Whereas when $\rho = (8\text{and}14) > \rho_{th}$, Fig. 6.6



$N = 1000; b = 1; L = 10; \beta = 0.0002; \mu = 0.001; \gamma = 0.002; \varepsilon = 0.0003; \omega = 0.0001; \alpha = 0.002; \lambda = 0.0005; r = 1.4$
$N = 1000; b = 1; L = 10; \beta = 0.0002; \mu = 0.001; \gamma = 0.003; \varepsilon = 0.0003; \omega = 0.0001; \alpha = 0.003; \lambda = 0.0005; r = 1.4$
$N = 1000; b = 1; L = 10; \beta = 0.0002; \mu = 0.001; \gamma = 0.0035; \varepsilon = 0.0003; \omega = 0.0001; \alpha = 0.004; \lambda = 0.0005; r = 1.4$

FIG. 6.4. *Plot between Infected and Recovered*



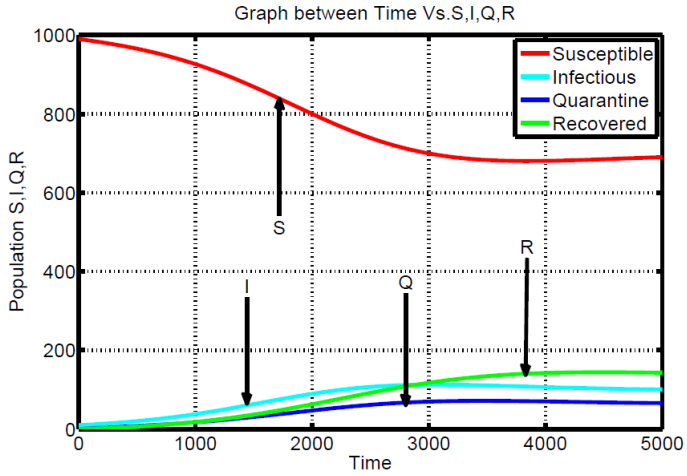
$N = 1000; b = 1; L = 10; \beta = 0.0002; \mu = 0.001; \gamma = 0.002; \varepsilon = 0.0003; \omega = 0.0001; \alpha = 0.002; \lambda = 0.0005; r = 1.4$
$N = 1000; b = 1; L = 10; \beta = 0.00024; \mu = 0.001; \gamma = 0.003; \varepsilon = 0.0002; \omega = 0.00014; \alpha = 0.0024; \lambda = 0.0004; r = 1.4$
$N = 1000; b = 1; L = 10; \beta = 0.00028; \mu = 0.001; \gamma = 0.0035; \varepsilon = 0.00025; \omega = 0.00018; \alpha = 0.003; \lambda = 0.0005; r = 1.4$

FIG. 6.5. *Plot between Susceptible and Recovered*

and 6.7 again shows that the trajectories converge to the endemic equilibrium and the simulation results are consistent with the theoretical analysis.

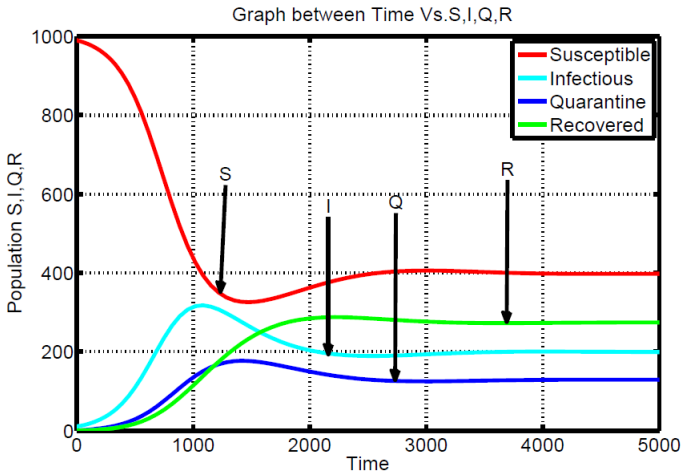
Node density also affects the connectivity of wireless sensor network. From Figs. 6.6 and 6.7 when node density increases connectivity increases and value of R_0 also increases. When the node density less than $\rho_{th} = 7.043116$ worm do not persist in the network. When the node density increases R_0 also increases and this is harmful for network.

6.3. Performance analysis of the model. Figs. 6.10, 6.11 and 6.12 shows the effect of parameters β , α and λ on the number of infectious nodes with change of values . As anticipated, when the value of β



$N = 1000; \rho = 8; b = 1; \beta = 0.0002; \mu = 0.001; \gamma = 0.002; \varepsilon = 0.0003; \omega = 0.0001 \alpha = 0.002; \lambda = 0.0005; r = 1$

FIG. 6.6. *Dynamic behavior of model when node density is 8.0*

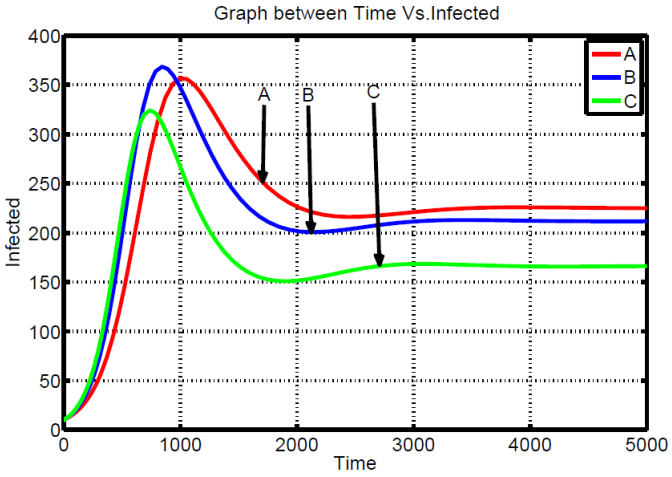


$N = 1000; \rho = 14; b = 1; \beta = 0.0002; \mu = 0.001; \gamma = 0.002; \varepsilon = 0.0003; \omega = 0.0001 \alpha = 0.002; \lambda = 0.0005; r = 1$

FIG. 6.7. *Dynamic behavior of model when node density is 14.0*

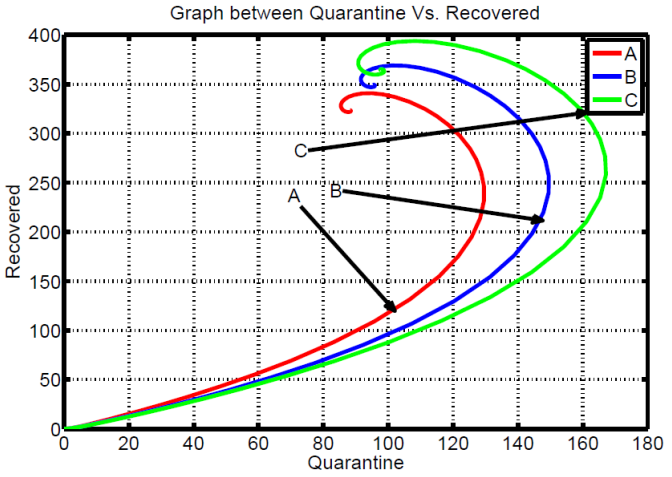
increased,the number of infectious nodes also increased. When λ increased and the remaining parameters are fixed number of recovered nodes increased. It is also found that,as time passes the number of infectious nodes increases and reached at maximum value at a certain stage this will depends on the parameters. Fig. 6.12 shows the comparison between [33] and proposed model. The plot shows that the proposed model is better in comparison to the existing model because more number of nodes are infected in existing model.

7. Conclusion. In this paper, we proposed an SIQRS model which describe the spread of worms in wireless sensor network. A controlling parameter R_0 is obtained,which played an important role to understand the dynamic behavior of worms propagation. The worms propagation can be predicted and eradicated from the system that depends on the value of R_0. We showed that the worm free and equilibrium is locally and globally asymptotically stable,when the value of reproduction number is less than or equal to one and endemic equilibrium is locally asymptotically stable when value of reproduction number is greater than one. Calculate the threshold value of communication radius and node density as well as establish the relation with reproduction



$N = 1000; \beta = 0.0003; \mu = 0.001; \gamma = 0.002; \varepsilon = 0.0003; \omega = 0.0001; \alpha = 0.004; \lambda = 0.0003; r = 1.4; \rho = 4.917$
$N = 1000; \beta = 0.00035; \mu = 0.001; \gamma = 0.0023; \varepsilon = 0.0003; \omega = 0.00014; \alpha = 0.0045; \lambda = 0.00034; r = 1.4; \rho = 4.917$
$N = 1000; \beta = 0.0004; \mu = 0.001; \gamma = 0.003; \varepsilon = 0.0003; \omega = 0.00018; \alpha = 0.005; \lambda = 0.0005; r = 1.4; \rho = 4.917$

FIG. 6.8. *Plot between Time and Infected nodes when node density is 8*



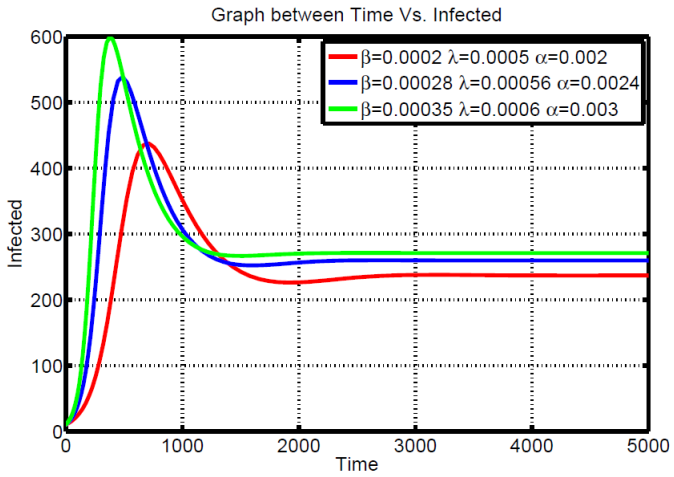
$N = 1000; \beta = 0.0003; \mu = 0.001; \gamma = 0.002; \varepsilon = 0.0003; \omega = 0.0001; \alpha = 0.004; \lambda = 0.0003; r = 1.4; \rho = 4.917$
$N = 1000; \beta = 0.0003; \mu = 0.001; \gamma = 0.002; \varepsilon = 0.0003; \omega = 0.00014; \alpha = 0.0045; \lambda = 0.0003; r = 1.4; \rho = 4.917$
$N = 1000; \beta = 0.0003; \mu = 0.001; \gamma = 0.003; \varepsilon = 0.0003; \omega = 0.00018; \alpha = 0.005; \lambda = 0.0003; r = 1.4; \rho = 4.917$

FIG. 6.9. *Plot between Time and Quarantine nodes when node density is 4.917*

number. Furthermore, we have done the simulation of proposed model with the help of MATLAB to verify and validate the results of proposed model. We study the impact of various parameters under different conditions. This model helps to developing an anti-virus mechanism for WSNs. Heterogeneous and moving nodes can be included for analysis of the model in future.On the basis of analysis worm riddance technique can be suggested.

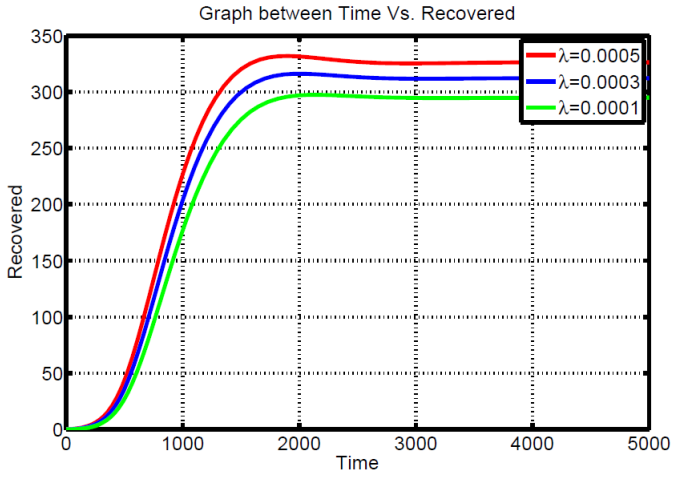
REFERENCES

[1] C.J.LIANG, R.MUSALOIU-E AND A.TERZIS, *Typhoon: A reliable data dissemination for wireless sensor networks,* in Proc. Wireless Sensor Networks,5th Eur. Workshop (EWSN 2008), Bologna, Italy, Jan. 2008.



$N = 1000; b = 1; L = 10; \beta = 0.0002; \mu = 0.001; \gamma = 0.002; \varepsilon = 0.0003; \omega = 0.0001; \alpha = 0.002; \lambda = 0.0005; r = 1.4$
$N = 1000; b = 1; L = 10; \beta = 0.00028; \mu = 0.001; \gamma = 0.002; \varepsilon = 0.0003; \omega = 0.0001; \alpha = 0.0024; \lambda = 0.00056; r = 1.4$
$N = 1000; b = 1; L = 10; \beta = 0.00035; \mu = 0.001; \gamma = 0.002; \varepsilon = 0.0003; \omega = 0.0001; \alpha = 0.003; \lambda = 0.0006; r = 1.4$

FIG. 6.10. *Plot between Time and Infected nodes with variation of parameters*



$N = 1000; b = 1; L = 10; \beta = 0.0002; \mu = 0.001; \gamma = 0.002; \varepsilon = 0.0003; \omega = 0.0001; \alpha = 0.002; \lambda = 0.0005; r = 1.4$
$N = 1000; b = 1; L = 10; \beta = 0.0002; \mu = 0.001; \gamma = 0.002; \varepsilon = 0.0003; \omega = 0.0001; \alpha = 0.002; \lambda = 0.0003; r = 1.4$
$N = 1000; b = 1; L = 10; \beta = 0.0002; \mu = 0.001; \gamma = 0.002; \varepsilon = 0.0003; \omega = 0.0001; \alpha = 0.002; \lambda = 0.0001; r = 1.4$

FIG. 6.11. *Plot between Time and Recovered nodes with variation of parameters*

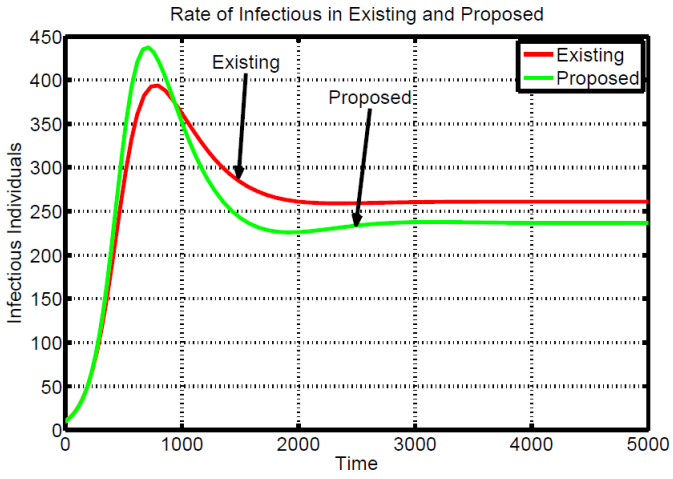
[2] S. FEDOR AND M. COLLIER, *On the problem of energy efficiency of multi-hop vs one-hop routing in Wireless Sensor Networks,* in Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops/Symposia (AINAW 07), pp. 380385, May 2007.

[3] H. SHI, W. WANG, AND N. KWOK, *Energy dependent divisible load theory for wireless sensor network workload allocation,* Mathematical Problems in Engineering, vol. 2012, Article ID 235289,16 pages,2012.

[4] J.ZHANG AND H.-N.LEE, *Energy-efficient utility maximization for wireless networks with/without multipath routing,* AEU International Journal of Electronics and Communications,vol. 64,no.2,pp. 99111,2010.

[5] H. CHEN, C. K. TSE, AND J. FENG, *Impact of topology on Performance and energy Efficiency in wireless sensor networks for source extraction,* IEEE Transactions on Parallel and Distributed Systems,vol.20,no.6,pp.886897,2009.

[6] C.Y. CHANG, J.-P. SHEU, Y.-C. CHEN, AND S.W. CHANG, *An obstacle-free and power-efficient deployment algorithm for wireless sensor networks,* IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans, vol.39,



$N = 1000; b = 1; L = 10; \beta = 0.0002; \mu = 0.001; \gamma = 0.002; \varepsilon = 0.0003; \omega = 0.0001; \alpha = 0.002; \lambda = 0.0005; r = 1.4$

FIG. 6.12. *Plot between Time Vs. Infected nodes For Existing and Proposed model*

no.4, pp. 795-806, 2009.

[7] M.H.KHOUZANI AND S.SARKAR, *Maximum damage battery depletion attack in mobile sensor networks,* IEEE Transactions on Automatic Control, vol. 56, no. 10, pp. 2358-2368, 2011.

[8] P. DE, Y. LIU, AND S.K. DAS, *An epidemic theoretic framework for vulnerability analysis of broadcast protocols in wireless sensor networks,* IEEE Transactions Mobile Computing, vol. 8, no. 3, pp. 413-425, 2009.

[9] S.ZANERO, *Wireless malware propagation:a reality check,* IEEE Security & Privacy, vol. 7, no. 5, pp. 70-74, 2009.

[10] SHENSHENG TANG, *A Modified SI Epidemic Model for Combating Virus Spread in Wireless Sensor Networks.* Int J Wireless Inf Networks (2011) 18:319326.

[11] J.L. SANDERS, *Quantitative guidelines for communicable disease control programs*, Biometrics, Vol. 27, pp. 883-893, 1971.

[12] R.PASTOR-SATORRAS AND A.VESPIGNANI, *Epidemic spreading in scale free networks*, Physical Letters, Vol. 86, pp. 3200-3203, 2001.

[13] Y. MORENO, M. NEKOVEE, AND A.VESPIGNANI, *Efficiency and reliability of epidemic data dissemination in complex networks*, Physical Review E, Vol. 69, pp. 1-4, 2004.

[14] A. KHELIL, C. BECKER, J. TIAN, AND K. ROTHERMEL, *Directed-graph epidemiological models of computer viruses.* In Proc. 5th ACM Intl Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems, pp. 54-60, 2002.

[15] H. ZHENG, D. LI, AND Z. GAO, *An epidemic model of mobile phone virus.* In 1st Intl Symposium on Pervasive Computing and Applications, pp. 1-5, 2006.

[16] S.K.TAN AND A.MUNRO, *Adaptive probabilistic epidemic protocol for wireless sensor networks in an urban environment.* In Proc. 16th International Conference on Computer Communications and Networks (ICCCN 2007), pp. 1105-1110, 2007.

[17] J. KIM, S. RADHAKRISHNAN, S.K. DHALL, *Measurement and analysis of worm propagation on Internet network topology*, Proc. of IEEE International Conference on Computer Communications and Networks, Chicago, Illinois, USA, pp.495-500, 2004.

[18] C.C. ZOU, W. GONG, D. TOWSLEY, *The monitoring and early warning for Internet worms*, IEEE Transactions on Networking, Vol. 13, No. 5, pp. 961-974, 2005.

[19] J.C. FRAUENTHAL, *Mathematical Modeling in Epidemiology*, Springer-Verlag, New York, USA,1980.

[20] D. MOORE, V. PAXSON, S. SAVAGE, C. SHANNON,S. STANOFORD ,N. WEAVER, *Inside the slammer worm*, IEEE Security & Privacy, Vol.1, No.4, pp. 33-39, 2003.

[21] R. DANTU, J.W. CANGUSSU, S. PATWARDHAN, *Fast worm containment using feedback contro*, IEEE Transactions on Dependable and Secure Computing, Vol. 4, No. 2, pp. 119-136, 2007.

[22] ZHANG SHU-KUI, GONG SHENG-RONG, CUI ZHI-MING, *Study On Spreading of Virus infection with SIRS characteristic in Wireless Sensor Network*, Proceeding of International Conference on Communication and Mobile Computing, 12-517, 2009.

[23] ROBERTO DI PIETRO, NINO VINCENZO VERDE, *Introducing Epidemic Models for Data Survivability in Unattended Wireless Sensor Networks*, Proceeding of 4th ACM Conference on Wireless Network Security (WiSec11), pp.11-22, 2011.

[24] WANG YA-QI, YANG XIAO-YUAN, *Virus spreading in wireless sensor networks with a medium access control mechanism.* Chin. Phys. B, 22(4), 040206, 2013.

[25] S.TANG, BRIAN L.MARK, *Analysis of Virus Spread in Wireless Sensor Networks: An Epidemic Model*, Design of Reliable Communication Networks pp. 86- 91, 2009.

[26] X.M. WANG AND Y. S. LI, *An improved SIR model for analyzing the dy namics of worm propagation in wireless sensor networks,* Chinese Journal of Electronics, vol. 18, no. 1, pp. 8-12, 2009.

[27] ARUN PRATAP SRIVASTAVA, SHASHANK AWASTHI, RUDRA PRATAP OJHA, PRAMOD KUMAR SRIVASTAVA AND SAURABH KATIYAR,

Stability Analysis of SIDR Model for Worm Propagation in Wireless Sensor Network, Indian Journal of Science and Technology, Vol 9(32),2016.

[28] Linhe Zhu and Hongyong Zhao, *Dynamical analysis and optimal control for a malware Propagation model in an information network*, Neurocomputing (2014).

[29] A. Martn del Rey, A.Hernndez Encinas, J.D.Hernndez Guilln, J.Martn Vaquero, A. Queiruga Dios and G. Rodrguez Snchez, *An Individual-Based Model for Malware Propagation in Wireless Sensor Networks*, DCAI, 13th Inter national Conference (2016),Advances in Intelligent Systems and Computing 474, pp. 223-230.

[30] Rudra Pratap Ojha, Pramod Kumar Srivastava, Shashank Awasthi and Goutam Sanyal, *Global Stability of Dynamic Model for Worm Propagation in Wireless Sensor Network*, Proceeding of International Conference on Intelligent Communication, Control and Devices (ICICCD 2016), Advances in Intelligent Systems and Computing Volume 479, pp.695-704, 2017.

[31] Shensheng Tang, David Myers and Jason Yuan, *Modified SIS epidemic model for analysis of virus spread in wireless sensor networks*, Int. J. Wireless and Mo bile Computing, Vol. 6, No.2, 2013, pp.99-108.

[32] Keshri, N.; Mishra, B.K., *Two time-delay dynamic model on the transmission of malicious signals in wireless sensor network*. Chaos Solitons Fractals, 68, 51158, 2014.

[33] Liping Feng, Liping Song, Qingshan Zhao, Hongbin Wang, *Modelling and Stability Analysis of Worm Propagation in Wireless Sensor Network*, Mathematical Problems in Engineering 2015.

[34] Driessche,P.V. and Watermouch, J, *Reproduction number and sub- threshold endemic equlibria for compartmental models of disease transmission*. Mathematical Biosciences, Vol. 180, pp. 29-48, 2001.

[35] LaSalle, J, *The stability of dynamical systems*. Regional to conference series in applied mathematics, pp. 96-106, 1976.

Edited by: Gulshan Shrivastava
Received: May 25, 2017
Accepted: Aug 23, 2017

DOI 10.12694/scpe.v18i3.1304
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 18, Number 3, pp. 243–252. http://www.scpe.org

PRIVACY ANALYSIS OF ANDROID APPLICATIONS: STATE-OF-ART AND LITERARY ASSESSMENT

GULSHAN SHRIVASTAVA AND PRABHAT KUMAR *

Abstract. In todays world, Android has revolutionized every facet of our lives. Almost all the important services such as banking, transportation, stock trade, medicine, education, etc. are extended to Android these days. Everything is available in the application market of android. Unfortunately, at the same time, the prosperity of these applications also attracts abusers and malicious attackers to perform different types of attacks. An appropriate action needs to be taken to protect Android device from these attacks. Android applications privacy analysis is an extension to the android privacy model, which traditionally emphasizes on prevention, and detection of attacks. It also deals with capturing, recording and analysis of Android applications to detect and investigate Android device intrusions. In this paper, we explore the comprehensive study about different techniques proposed to perform Android applications analytics. In addition to this, various aspects of android applications analytics have been reviewed along with related technologies and their limitations. This gives enhanced recognition of the problem, existing explanation space, and potential research scope to analyze and investigate various Android device intrusions against such attacks efficiently.

Key words: Android Analytics, Smartphone intrusion, Android attack

AMS subject classifications. 68M12, 68M14

1. Introduction.

When a user download application in an Android device from Android market (Google Play, 1Mobile, Todd etc.) and try to install (use application in advanced version of Android) on device, that application require some permissions from user to use Android device functionality. It is the wish of user to grant those permissions or not. Recent studies have shown that majority of users does not bother about this permission due to lack of knowledge about it and use it directly; at that stage user is not aware about the interruption. Even if any user wants to read these permissions, but these permission descriptions are too confusing and tough to understand [1, 2].

Privacy is a serious concern because these applications often ask for the access to confidential and sensitive information on Android device to work properly. Twitter, for example, asks to record audio with the microphone (at all time without user confirmation), read Google service configuration, read phone status and identity, etc. A recent research explored, 96% of applications need EMAIL, 92% needs ADDRESS_BOOK, 84% needs LOCATION, 52% need CAMERA, and 32% need CALENDAR permissions [4]. Many social networking websites like Facebook, Twitter and Linkedin have lots of user and they use advertisements for instance sponsored ads and posts, to generate good revenue. To get the significance and accomplishment of their advertisements i.e., targets advertisements, such website collect users private information through their applications demanding unnecessary permissions. As a conservatory of current research on permissions, we examine end users experience regarding the level of permissions being requested by various Android applications.

1.1. Android as a Major Application Provider.

In between August 2010 to May 2016, Android had over 65 billion application downloads. Also Apple and Google store combined revenue is $2.23 billion in the US [26]. Android application is not pre-screened for quality even in its official market. AppBrain gives a report that 33% applications are rated low quality based on its rating and recommendation. In 2011, Juniper Networks reported that 47.2% Android malware samples increased in between July and November 2011 [27]. Some reputed organization like McAfee [28] and Kaspersky Lab [29] also reported for continued exploits. The intention of these malware is to collect the users personal data and share or sell them to targeted advertisement companies.

1.2. Motive of Amendment in Android Application Display.

Permissions are important part of an application but they are only displayed at the time of installation. Therefore, Android has done some amendment in their policy to display application permission. As per Android 6.0, whenever an application

*Department of Computer Science and Engineering, National Institute of Technology Patna, INDIA({gulshanstv@gmail.com, prabhat@nitp.ac.in}).

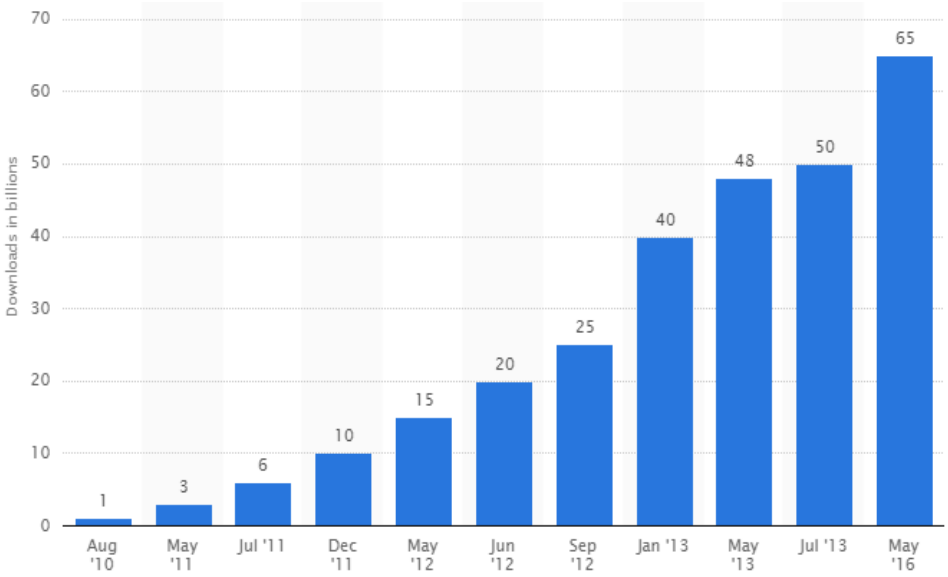


FIG. 1.1. *Google Play: number of Android app downloads 2010-2016 [26]*

requires any permission then it will only ask for that particular permission, instead of these lifetime grants of permissions.

1.3. Privacy on Android Permission. Android application permissions asked by an application are listed in AndroidManifest.xml. As per many research survey [35, 36] it has been found that very less number of users understand that what these permissions actually asking for. These surveys are based on large risk assessment of users about security and privacy risks. Felt et al. [37] also did survey about the way of asking this permission and presented some set of guidelines for user to take into account while making decision about their privacy and security concerns. King et al. [38] presented the users expectation from their Smartphone for entire usage. In this work, they collected interview results of iPhone and Android users and highlighted the problem faced by them at the time when application and web links, gathers the information of possible privacy faults, and process to select the best application from app markets.

The paper is organized as follows: Section II presents Literature review. In Section III, we explore the Android applications permission and Android application mechanism in detail. Section IV discusses about the limitation of the Android permission. In Section V, we describe the permission analytics. Finally, Section VI concludes the paper.

The paper is organized as follows: Section II presents Literature review. In Section III, we explore the Android applications permission and Android application mechanism in detail. Section IV discusses about the limitation of the Android permission. In Section V, we describe the permission analytics. Finally, Section VI concludes the paper.

2. Literature Review. As Android device usage is increasing day by day, at the same time the risk factors of security breach are also increasing. The biggest challenge is to identify and prevent those risks. Malicious Android applications analysis and detection is admired research fields as per escalation in usage of Android market. While Android has survived openly since 2008, a major amount of effort has been conduct on studying the Android permissions/security model. A lot of focus have been done on creating hypothetical formalizations of Android security mechanism or presenting enhancements to system security, and is mainly out of scope.

In Android 4.3, hidden features like App Ops allow the user for revoke the irrelevant permissions as per application but as per the unawareness user do not explore it. According to Verizons 2015 data, in middle attack or applications that store sensitive information insecurely or send data incomplete. According to the Jupiter network report [11, 12], 76% of mobile users depend on their mobile devices to access their most sensitive

personal information, such as online banking or personal medical information. This trend is even more noticeable with those who also use their personal mobile devices for business purposes. Nearly nine out of ten business users report that they use their own mobile devices to access sensitive work-related information. It has been reported by the Junipers Network Mobile Threat Centre that in 2011, there was an unparalleled increase in Android malware attacks with an increase of about 155% from previous years in all platforms.

According to Googles CEO, 1.5 billion known users use Android. Recent version used is Android 6.0 Marshmallow. Around 8 out of 10 are Android users. In addition, 1.6 million devices access Google play. The SAN report analysis outsourcing growth to 1.4 million in 2016. Even in the latest edition of Android operating system, there is no provision to handle and manage security and privacy issues. Google Play provides security to play store and manage its privacy concern security.

Researchers have to be very conscious about permission usage in Android. Kelley et al. [2] explained about unawareness of permissions. Felt et al. [9] explored that Android failed to clearly inform their users about permissions. Kelley et al. [6] proposed a privacy checklist for the privacy risk of an application, they also showed that proposed checklist significantly affected user for application selection. Felt et al. [1] explored about gap analysis between free and paid applications in the frequency of dangerous permission being requested after surveying 100 paid & 856 free applications; for instance, ratio of free and paid is 7:2 in the aspect of INTERNET permission. Hypothesis of [7] presents the frequency of request from free applications for INTRNET permission for advertisements. Felt et al. [8] fixed some permission that are common between normal, dangerous and signature permissions by Google but still behave like dangerous permission legitimately. Felt et al. [9] concluded that 1/3 of 940 applications are measured as over privileged. Vidas et al. [10] proposed a static analysis tool to enhance the set of permissions and minimize it so that application behaves correctly. This literature mainly focuses on malicious analysis and detection based on Android applications permission. Enck et al. [30], TaintDroid has filled the distance between system security and user permissions, focusing on analysis of application, which request for permissions through permission list. Vidas et al. [31] gives details about application permissions requests, finding ubiquitous permissions creep, because of existing developer APIs that composes it making it complicated for developers to align their permission requirements with application functionality. Felt et al. [32] endeavor to make clear permissions to developers. However, neither of these articles investigates end users considerate for permissions.

Researchers have exposed, attack vectors for applications to compose permission requirements that are not detailed to users. Others who have come across at Android permissions have attempted to collect applications that necessitate similar permissions to simplify the existing method or have attempted an evaluation of the dissimilarity between current Android permission systems [34]. Haris et al. [13] concentrated on the risk in mobile computing and privacy leaks in mobile connectivity and mobile sensing. State-of-art is similar to [3, 5] for multiple mobile platforms. Crussell et al. [14] proposed DNADroid that approach used program dependency graphs i.e. detects the similar applications. This approach generates a low false positive rate. For the increase in the system performance, they evaluate application against similar names. Zheng et al. [15] proposed a static analysis framework i.e. DroidAnalytics. It detected obfuscated Android malware. Signature generation performed in three level methods. Repackaged malware detected by the class and application level in that sequence. Then it is compared to other effective obfuscated technique such as method renaming, further control flow goes to-obfuscation and string encryption. In this article, authors didnt discuss about the time taken to analyze the applications. Hanna et al. [17] proposed the Juxtapp, they introduced the Feature Hashing for the detection of similar applications. It is resilient to some amount of obfuscation. They used obfuscation resilience of algorithms to detect the repackaging apps.

Zhauniarovich et al. [33] proposed detection of repackaged method that showed the results of similar techniques involved in code analysis. This approach is based on the content of. apk file. Repackaged applications have minor changes in lots of file, which is difficult to identify. Zhou et al. [16] proposed DroidMOSS method to repackaging detection through Fuzzy Hashing. In this paper, authors generate a hash to perform divide and conquer method for that application. This proposed system is robust as claimed by author but it is based on number of chunks, which is subpart of an application. In another research, Zhou et al. [18] proposed AppINK based on client side for repackage detection. In this method, watermarking technique is used for authentication of an application. Based on watermarking, user can identify the repackaged application. Authors validate their

Android Application	Category	Permission Requested	Flow Permission
My Calendar	Productivity	STORAGE	PHONE NO → NETWORK
		LOCATION	
		NETWORK	
		PHONECALLS	
My Space	Social	STORAGE	IMEI N →NETWEOK
		NETWORK	
		PHONECALLS	
Gmail	Communication	NETWORK	STORAGE→NETWORK
		STORAGE	

FIG. 3.1. *Android Applications and their Permissions Examples*

proposed method with the help of some pattern matching algorithm. Wu et al. [19] provided the novel policy that deals with repackaged issues. This is dynamic event flow and verifies the applications in the future, by the help of watermark injected in the applications. The weakness of this techniques is it takes more overhead and inherently limited resources on Android device when identify the watermark from applications. It is not sufficient automatic watermark extraction technique.

Kang et al. [20] designed fast malware detector by using creator information. In this detection is done according to malware permission and behavior. This is based on the static analysis that is used to certify serial number and detects the malware. It is light weighted process, which analyze particular part of applications according to malware behavior and permission. This provides the very high detection accuracy.

Enck et al. [21] explained about Android security mechanism based on permissions and security issues for inter-communication between apps. Inter-Component Communication (ICC) and Intents have not been explored the way permissions have been investigated. Most of the existing ICC based studies focuses on finding the ICC related vulnerabilities. The work depicted in [22] investigated the IPC framework and interaction of system components. ComDroid [23] detects the ICC related vulnerabilities. The work depicted in [24] suggested improvement in ComDroid by segregating the communication messages into inter and intra-application groups so that the risk of inter-application attacks may be reduced. The work depicted in [25] performs information flow analysis to investigate the communication exploits.

3. Android Application Permission. Permission system is a very basic security aspect implemented in todays Android devices. The Android Market [43] has experienced a great boon as it allows anyone to upload applications to the market. By which, a developer can be rewarded with a monetary gain by receiving ad revenue or by charging for the services provided by the application. Permissions are provided for security mechanism to defend the receptive APIs thus to be utilized only by trusted applications. Before installing an application, a list of permissions is requested to the user. The permissions are granted on all or nothing basis i.e. a user is not allowed to selectively allow or disallow permissions that an application is requests.

There are mainly four types of Android permissions:

1. Signature
2. System
3. Normal
4. Dangerous

Signature and System are reserved for the application that comes preinstalled on the firmware or have a key signed in firmware. These types of permissions are not available to third party applications. When an application is reinstalled, it should be signed by the matching certificate as the previous application in order to obtain the old applications data, and then only Signature permission will be granted by the system without user interaction.

Normal and Dangerous are other most important type of permissions. Normal permissions are low level of permission which do not required any user grant. User can examine this permission and if found any doubt then no other option rather than un-installation of that particular application is available. Whereas dangerous

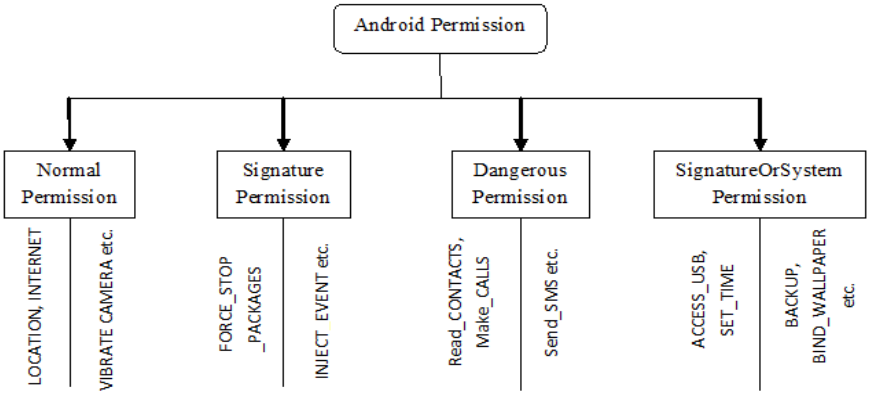


FIG. 3.2. *Categorization of Permission*

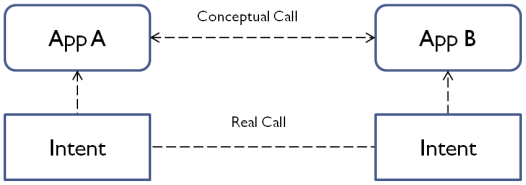


FIG. 3.3. *Inter Process Communication*

permission, are asked by user and it is the user who wish to grant them or not. If a user grants any such type of permission, then it is permanently granted. These types of permission are mandatory to display from application developer at the time of installation of that application.

3.1. Inter-process Communication. Two ways process communication can be divided into two categories firstly by using any of the conventional UNIX-type mechanisms for instance, the file system, local sockets, or signals etc. Whenever an application directly communicates with another application without user interaction it shows, its conceptual call but that time it uses intent and intent filter in real call as fig.4. Whenever a component is requested by an application, the Android system checks if the corresponding process is running and the component is instantiated or not. If either of them is not available, it is created by the system.

Android also provides IPC mechanisms:

1. **Binder:** It is a frivolous remote process call system based on compatibility. It is intended for elevated performance while executing process calls.
2. **Intents:** It represents an "intention" to obtain something, which is finished. For instance, if application needs to demonstrate a web page, its intent to view the URL is expressed by creating and sending an Intent instance to the system. The system finds corresponding code (for this example Browser) which knows to offer service to that Intent, and executes it as shown in figure 5. Intents are also used to broadcast system-wide events. Intents are capable of crossing process boundaries and can transfer information between applications.

   Types of Intent:

   (a) **Explicit Intent**

```
String url = www.google.com;
Intent explicit = new Intent(Intent.ACTION_VIEW);
explicit.setData(Uri.parse(url));
explicit.setPackage(com.Android.chrome);
startActivity(explicit);
```

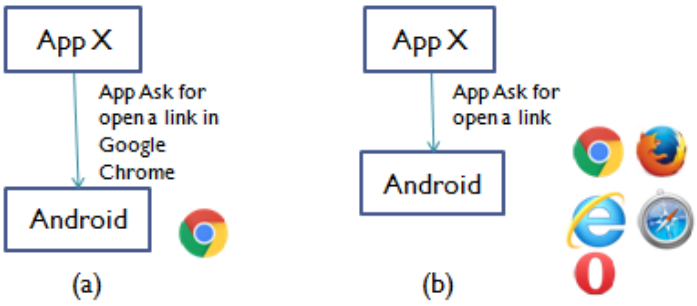


FIG. 3.4. *(a) Explicit Intent; (b) Implicit Intent*

(b) **Implicit Intent**

```
String url = www.google.com;
Intent implicit = new Intent(Intent.ACTION_VIEW);
implicit.setData(Uri.parse(url));
startActivity(implicit);
```

3. **Content Providers:** ContentProvider serves as a data storage, which provides access to data stored on the device; for example, user's list of contact scan can be thought of as ContentProvider. An application can access data provided by other application via ContentProvider, and an application creates a ContentProviders to share its data.

3.2. Android Permission Mechanism. The middleware layer of Android architecture provides permission mechanism. To implement mandatory access control (MAC) on inter-communication calls, reference monitor is used. Standard Android permissions such as INTERNET, PHONE_CALLS, and SEND_SMS etc. are used to protect security sensitive interfaces. Application holds these permissions to be capable of making phone calls, to access the Internet or to send text SMS. Apart from that, application can declare different types of permission labels to bar access to its interfaces. Necessitated permissions are displayed in a file and approved during installation based on user confirmation. When an application requests inter-communication calls, reference monitor checks whether the application has required permissions or not.

4. Limitations. The limitations that lead to weaknesses, of Android security model are summarized as follows:

1. Limitation 1: Android uses an install-time permission model:
   (a) The install-time permission works on an all-or-nothing principle: in older versions, a user has to grant all permissions requested by an application, in order to proceed for installation.
   (b) Use is mostly unaware of consequences of requested permissions, which leads to approval of permissions [8].
   (c) Application developers try to expand their requested permissions, many of which are not necessary for applications execution. [40]
   (d) Lack of sufficiently fine-grained permissions. [41]
   (e) There is no provision of runtime permission revocation, and control [42].

   Once the application is installed on a device, the application is authorized to use the device, as it wants to use all its granted permissions. There are no controls to limit the uses of permissions.

2. Limitation 2: In comparison to desktop environment, such as Windows and UNIX executables, Android applications are easier to reverse engineer. This is possible due to presence of hardware-independent Dalvik bytecode files. These files contain great amount of data of original Java sources.

3. Limitation 3: Lack of isolation mechanism for third-party libraries contained within applications [43].

Due to this, third-party libraries cannot be stopped from misusing the granted permissions of host applications. Conversely, ill-behaving host applications may tamper with the libraries, for example, by performing improper advertisement display or click fraud.

4. Limitation 4: Android lacks a configurable, runtime ICC control for the following purposes:
   (a) To prevent an application from accessing any open interfaces of another application, despite the former having obtained the required permissions at its install time [23, 9]. Combined with weakness 5 below, this lack of runtime inter-application access control can lead to data leakage and confused deputy problems.
   (b) To prevent an application from intercepting an intent broadcast and possibly stopping its propagation afterwards [23]. By intercepting system-event broadcasts, a malicious application is able to stealthily intercept important system events that contain sensitive information, such as an incoming call or SMS.
   (c) To isolate applications and prevent them from communicating via ICC and other shared channels [44]. The presently unrestricted ICC among applications in Android can be exploited by colluding applications.

5. Limitation 5: Application developers could be unfamiliar with the subtle aspects of Android ICC [45], which may lead to unintentional exposure of applications private interfaces and data.

5. Permission Analytics. An accurate estimate of success matrix is one of the most basic requirements for Android based organization and researchers planning for analytics to get ultimate goal of identified risk and satiate this need of Android users respectively [39]. Analytics have moved on from data analytics to all other digital media and hence renamed accordingly. For example, Web Analytics for data on web, Mobile Analytics for mobile applications, Multi-Channel Analytics for collecting analytical data across heterogeneous platforms, Social Media Analytics for banking, Facebook Insights, Twitter Analytics and YouTube Analytics.

To maintain long term sustainable relationship with customers, rather than being content with short-term boost in the conversion rate by giving priority to customer over an experts or analysts view helps in decreasing the bounce rate and optimize global maxima rather than local maxima for every search, by focusing on key performance parameters (KPIs) for success and failure. These new age analytics that intend data from each achievable medium including web and social site is helping business syndicates to access even the most personal information about their customers and prospect research aims in managing of disclosure of personal information and giving only those aspects which could amplify their breach investigation, reports that 0.03% of Android device per week infected by the malicious code out of 10 million of mobile device. The Applications vulnerability is consisted of risky applications prone in man customer base. Permission analytics are categorized based on their main approaches as follows.

1. **Static Analysis**: A static analyzer performs inspection of application without its execution; analyzer is able to achieve high code coverage. Since it have been analyze the applications complete source or recovered code. However, it does not say anything about execution context as well as its execution. Apart from that, it faces difficulties in the presence of dynamic code loading and code obfuscation [46]. Analyzer is required to perform accurate static analysis on an event-driven Android application to deal with the various Android-specific issues such as: [47]:
   (a) Presence of multiple entry points in application;
   (b) App components may run asynchronously
   (c) Possibility of multiple callbacks because of applications lifecycle states, as well as listeners to system and GUI events;
   (d) Application might accept both intra and inter application ICC. In addition, Java reflection and native code possess additional challenges [48].

2. **Dynamic Analysis:** It is performed by executing the application. This can be done using either real or virtual environment for instance, Android Virtual Device (AVD) and monitoring the application while

it is executing. Huge variety of dynamic analysis techniques are given in the literature. Performing dynamic analysis on Android applications faces new difficulties such as Binder-based ICC, Androids managed resources and event triggers [49].

3. **Static & Dynamic Analysis:** Due to limitations of static and dynamic analysis techniques individually, many systems choose to perform both: static and dynamic analysis to complement each other. Google Play uses Bouncer, which scans for malicious applications automatically. Many security researchers have made attempts to figure out working of bouncer, as it is not public [50, 51]. According to Oberheide et al. [54] seemed to perform dynamic analysis on an application by running it on a QEMU-based emulated Android environment for only 5 minutes. Mobile-Sandbox [59] also employs both static and dynamic analysis. Application requested permissions are analyzed for static analysis. Then, the applications Dalvik bytecode is converted to small code by using baksmali5. During dynamic analysis stage; to improve code coverage information on timers and broadcasts are also collected, Mobile-Sandbox stores runtime information at the three levels:
   (a) Dalvik level monitoring using TaintDroid and a customized version of DroidBox;
   (b) Native code monitoring using ltrace; and
   (c) Network traffic monitoring into a PCAP file.

   Mobile-Sandbox utilizes the monkey runner *tool6To* simulate user interaction, to trigger potentially malicious behavior external events, such as incoming calls or SMS messages, are also simulated.

4. **Analyzing Application Metadata:** Metadata is a source of useful information that can be used to infer the applications capability, and therefore potential behavior. In order to access security risks, Metadata of untrusted applications is analyzed in several works. WHYPER [52] is a Natural Language Processing (NLP) based system, which uses textual application description of an application and permission list to see if application information justifies permission lists. Although WHYPER is shown to achieve substantial improvement in accuracy over keyword-based search, numerous challenges still exist for it to give a dependable, automated application risk estimate.

The mobile analysts have done many analyses for Android applications but permission analysis is done only for hypothetical analysis and to get its accuracy of risk. Many researchers state that permission based privacy analysis is not to be done by Android application developers [53].

6. Conclusion. In this paper, we have described the Android application permissions and its mechanism in detail. This paper also presented the weakness of permissions also various aspects of Android applications analytics have been reviewed along with related technologies and their limitations. We have provided better way to understand the problem that may occur in research solution and the future scope of analyzing the various Android device permissions against various risks.

REFERENCES

[1] Felt, A. P., Ha, E., Egelman, S., Haney, A., Chin, E., & Wagner, D., *Android permissions: User attention, comprehension, and behavior* (2012, July). In Proceedings of the eighth symposium on usable privacy and security (p. 3). ACM.

[2] Kelley, P. G., Consolvo, S., Cranor, L. F., Jung, J., Sadeh, N., & Wetherall, D., *A conundrum of permissions: installing applications on an Android smartphone* (2012, February). In International Conference on Financial Cryptography and Data Security (pp. 68-79). Springer Berlin Heidelberg.

[3] La Polla, M., Martinelli, F., & Sgandurra, D., *A survey on security for mobile devices* (2013). IEEE communications surveys & tutorials, 15(1), 446-471.

[4] Leyden, J., *The TRUTH about LEAKY, STALKING, SPYING smartphone applications* (2013). The Register.

[5] Suarez-Tangil, G., Tapiador, J. E., Peris-Lopez, P., & Ribagorda, A.,*Evolution, detection and analysis of malware for smart devices* (2014). IEEE Communications Surveys & Tutorials, 16(2), 961-987.

[6] Kelley, P. G., Cranor, L. F., & Sadeh, N., *Privacy as part of the app decision-making process* (2013, April). In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 3393-3402). ACM.

[7] Barrera, D., Kayacik, H. G., van Oorschot, P. C., & Somayaji, A., *A methodology for empirical analysis of permission-based security models and its application to Android* (2010, October). In Proceedings of the 17th ACM conference on

Computer and communications security (pp. 73-84). ACM.
[8] Felt, A. P., Greenwood, K., & Wagner, D., *The effectiveness of application permissions* (2011, June). In Proceedings of the 2nd USENIX conference on Web application development (pp. 7-7).
[9] Felt, A. P., Chin, E., Hanna, S., Song, D., & Wagner, D, *Android permissions demystified* (2011, October). In Proceedings of the 18th ACM conference on Computer and communications security (pp. 627-638). ACM.
[10] Vidas, T., Christin, N., & Cranor, L, *Curbing Android permission creep* (2011, May). In Proceedings of the Web (Vol. 2, pp. 1-5).
[11] *Trusted mobility Index* (2012, May), Retrieved from `http://www.juniper.net/us/en/local/pdf/additional-resources/7100155-en.pdf.`
[12] *2011 Mobile Threats Report* (2012, Feb.), Retrieved from `https://www.juniper.net/us/en/local/pdf/additional-resources/jnpr-2011-mobile-threats-report.pdf`
[13] Haris, M., Haddadi, H., & Hui, P., *Privacy leakage in mobile computing: Tools, methods, and characteristics* (2014). arXiv preprint arXiv:1410.4978.
[14] Crussell, J., Gibler, C., & Chen, H.,*Attack of the clones: Detecting cloned applications on Android markets* (2012, September). In European Symposium on Research in Computer Security (pp. 37-54). Springer Berlin Heidelberg.
[15] Zheng, M., Sun, M., & Lui, J. C., *Droid analytics: a signature based analytic system to collect, extract, analyze and associate Android malware* (2013, July). In Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on (pp. 163-171). IEEE.
[16] Zhou, W., Zhou, Y., Jiang, X., & Ning, P., *Detecting repackaged smartphone applications in third-party Android marketplaces* (2012, February). In Proceedings of the second ACM conference on Data and Application Security and Privacy (pp. 317-326). ACM.
[17] Hanna, S., Huang, L., Wu, E., Li, S., Chen, C., & Song, D., *Juxtapp: A scalable system for detecting code reuse among Android applications* (2012, July). In International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (pp. 62-81). Springer Berlin Heidelberg.
[18] Zhou, W., Zhang, X., & Jiang, X., *AppInk: watermarking Android apps for repackaging deterrence* (2013, May). In Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security (pp. 1-12). ACM.
[19] Wu, D. J., Mao, C. H., Wei, T. E., Lee, H. M., & Wu, K. P.,*Droidmat: Android malware detection through manifest and api calls tracing* (2012, August). In Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference on (pp. 62-69). IEEE.
[20] Kang, H. J., Jang, J. W., Mohaisen, A., & Kim, H. K., *Androtracker: Creator information based Android malware classification system* (2014). In Information Security Applications-15th International Workshop, WISA (Vol. 8909).
[21] Enck, W., Ongtang, M., & McDaniel, P., *Understanding Android security* (2009). IEEE security & privacy, 7(1), 50-57.
[22] Enck, W., Ongtang, M., & McDaniel, P., *On lightweight mobile phone application certification* (2009, November). In Proceedings of the 16th ACM conference on Computer and communications security (pp. 235-245). ACM.
[23] Chin, E., Felt, A. P., Greenwood, K., & Wagner, D.,*Analyzing inter-application communication in Android* (2011, June). In Proceedings of the 9th international conference on Mobile systems, applications, and services (pp. 239-252). ACM.
[24] Kantola, D., Chin, E., He, W., & Wagner, D., *Reducing attack surfaces for intra-application communication in Android* (2012, October). In Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices (pp. 69-80). ACM.
[25] Perkins, J., & Gordon, M., *DroidSafe* (2016). Massachusetts Institute of Technology Cambridge United States.
[26] *Statista* 2017. Cumulative number of apps downloaded from the Google Play as of May 2016.
[27] Vennon, T., & Center, J. G. T.,*Mobile malware development continues to rise, Android leads the way* (2016). In FAST.
[28] *McAfee*, S. C. CA, USA.(2011). Mcafee Threats Report: Third Quarter 2011.
[29] Namestnikov, Y, *It threat evolution* Q3 2011, 2011.
[30] Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B. G., Cox, L. P., & Sheth, A. N., *TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones* (2014). ACM Transactions on Computer Systems (TOCS), 32(2), 5.
[31] Vidas, T., Christin, N., & Cranor, L., *Curbing Android permission creep* (2011, May). In Proceedings of the Web (Vol. 2, pp. 1-5).
[32] Felt, A. P., Egelman, S., & Wagner, D., *I've got 99 problems, but vibration ain't one: a survey of smartphone users' concerns* (2012, October). In Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices (pp. 33-44). ACM.
[33] Zhauniarovich, Y, *Security of the Android operating system.* In International Conference on Risks and Security of Internet and Systems (pp. 272-274). Springer International Publishing.
[34] Au, K. W. Y., Zhou, Y. F., Huang, Z., Gill, P., & Lie, D, *Short paper: a look at smartphone permission models* (2011, October). In Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices (pp. 63-68). ACM.
[35] Geneiatakis, D., Fovino, I. N., Kounelis, I., & Stirparo, P.,*A Permission verification approach for Android mobile applications* (2015). Computers & Security, 49, 192-205.
[36] Lockheimer, H., *Android and security* (2012). Google Mobile Blog, Feb, 2.
[37] Fragkaki, E., Bauer, L., Jia, L.,& Swasey, D., *Modeling and enhancing Androids permission system* (2012, September). In European Symposium on Research in Computer Security (pp. 1-18). Springer Berlin Heidelberg.
[38] King, J, *How Come I'm Allowing Strangers to Go Through My Phone? Smartphones and Privacy Expectations.*

[39] Google play (2017), Retrived from https://en.wikipedia.org/wiki/Google_Play
[40] Ongtang, M., McLaughlin, S., Enck, W., & McDaniel, P.,*Semantically rich applicationcentric security in Android* (2012). Security and Communication Networks, 5(6), 658-673.
[41] Jeon, J., Micinski, K. K., Vaughan, J. A., Fogel, A., Reddy, N., Foster, J. S., & Millstein, T., *Dr. Android and Mr. Hide: fine-grained permissions in Android applications* (2012, October). In Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices (pp. 3-14). ACM.
[42] Hornyack, P., Han, S., Jung, J., Schechter, S., & Wetherall, D, *These aren't the droids you're looking for: Retrofitting Android to protect data from imperious applications* (2011, October). In Proceedings of the 18th ACM conference on Computer and communications security (pp. 639-652). ACM.
[43] Pearce, P., Felt, A. P., Nunez, G., & Wagner, D. *Addroid: Privilege separation for applications and advertisers in Android* (2012, May). In Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security (pp. 71-72). Acm.
[44] Bugiel, S., Davi, L., Dmitrienko, A., Fischer, T., Sadeghi, A. R., & Shastry, B., *Towards Taming Privilege-Escalation Attacks on Android* (2012, February). In NDSS (Vol. 17, p. 19).
[45] Poeplau, S., Fratantonio, Y., Bianchi, A., Kruegel, C., & Vigna, G., (2014, February), *Execute This: Analyzing Unsafe and Malicious Dynamic Code Loading in Android Applications.* In NDSS (Vol. 14, pp. 23-26).
[46] Arzt, S., Rasthofer, S., Fritz, C., Bodden, E., Bartel, A., Klein, J., & McDaniel, P., *Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps* (2014). Acm Sigplan Notices, 49(6), 259-269.
[47] Grace, M., Zhou, Y., Zhang, Q., Zou, S., & Jiang, X., *Riskranker: scalable and accurate zero-day Android malware detection* (2012, June). In Proceedings of the 10th international conference on Mobile systems, applications, and services (pp. 281-294). ACM.
[48] Zhang, Y., Yang, M., Xu, B., Yang, Z., Gu, G., Ning, P., & Zang, B.,*Vetting undesirable behaviors in Android apps with permission use analysis* (2013, November). In Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security (pp. 611-622). ACM.
[49] Oberheide, J., & Miller, C.,*Dissecting the Androids Bouncer.* (2013, November) SummerCon.
[50] Percoco, N. J., & Schulte, S., *Adventures in bouncerland: failures of automated malware detection with in mobile application markets* (2012). Black Hat USA, 12.
[51] Spreitzenbarth, M., Freiling, F., Echtler, F., Schreck, T., & Hoffmann, J., *Mobile-sandbox- having a deeper look into Android applications* (2013, March). In Proceedings of the 28th Annual ACM Symposium on Applied Computing (pp. 1808-1815). ACM.
[52] Pandita, R., Xiao, X., Yang, W., Enck, W., & Xie, T., *WHYPER: Towards Automating Risk Assessment of Mobile Applications* (2013, August). In USENIX security (Vol. 13, No. 20).
[53] Tan, D. J., Chua, T. W., & Thing, V. L., *Securing Android: a survey, taxonomy, and challenges.* ACM Computing Surveys (CSUR), 47(4), 58.
[54] Oberheide, J., & Miller, C, *Dissecting the Androids Bouncer.* SummerCon, 2012.

Edited by: Kavita Sharma
Received: May 24, 2017
Accepted: Sep 1, 2017

DOI 10.12694/scpe.v18i3.1305
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 18, Number 3, pp. 253–260. http://www.scpe.org

A NOTE ON CORRECTLY GATHERING RESULTS FROM JR'S CONCURRENT INVOCATION STATEMENT

RONALD A. OLSSON*

Abstract. JR's concurrent invocation statement (*CoStmt*, for short) provides a form of collective communication. The thread executing a *CoStmt* can gather results from each of many invocations. This paper presents a problem that arises in using Java `List` structures in gathering such results. It presents incorrect and correct attempts at solutions and discusses the tradeoffs, including performance, among the correct solutions.

Key words: Concurrency, concurrent programming, JR programming language, concurrent invocation, synchronisation

AMS subject classifications. 68N15, 68N19, 68N20

1. Introduction.

1.1. Background. The JR concurrent programming language extends Java with a richer concurrency model [1, 2], which is based on that of SR [3, 4]. JR provides dynamic remote virtual machine creation, dynamic remote object creation, remote method invocation, dynamic process (thread) creation, rendezvous, asynchronous message passing, semaphores, concurrent invocation, and shared variables. These language features are provided by adding several new types and statements to the Java language. JR provides fairly transparent distributed programming (built using RMI, but hiding its details) and automatic program termination when the program has been detected to have quiesced. The JR implementation [2, 5] extends the Java implementation.

This paper focuses on an interesting problem that arises with the use of JR's concurrent invocation statement (*CoStmt*, for short) [5, 6]. The *CoStmt* can be viewed as providing collective communication between one thread and a group of others, for example, in applications such as initiating subparts of numerical computations, reading/writing a distributed database, and distributed voting schemes. Several concurrent programming languages and systems provide mechanisms to facilitate such communication, such as: MPI's [7] collective communication; and .NET's [8] delegates and asynchronous invocations. The general pattern is that one thread broadcasts data to a group of threads and then gathers back results from each in the group.

1.2. The problem and its challenges. This problem arose in practice when we were converting part of a JR program with a *CoStmt* from using an array to use an `ArrayList`. The specific problem arises when the thread gathering results of concurrent invocation needs to save the results in such a way as to preserve their order, which is fairly typical use of a *CoStmt*. Gathering into an array solves this problem easily, using the natural order provided by the array's subscripts. Gathering into a `List`, however, turns out to be more challenging. Code that gathers into a `List` that appears to be similar to code that gathers into an array does not always work. On the surface such code appears as though it should work, but these appearances are misleading.

Part of the problem involves issues of synchronisation in updating a `List`. Additional synchronisation can, in some cases, make a solution work, but it turns out that one correct solution requires no such additional synchronisation.

1.3. Outline. Section 2 introduces the *CoStmt* via examples that gather results into an array. It then presents attempts—incorrect and correct—for equivalent *CoStmts* using various kinds of Java `List` structures and discusses tradeoffs among the correct solutions, including the amount of required synchronisation. Section 3 describes the performance of the correct solutions and further explores the costs of synchronisation in the underlying Java `List` structures. Finally, Sect. 4 concludes the paper.

* Department of Computer Science; University of California, Davis; One Shields Avenue; Davis, CA 95616-8562 USA; (olsson@cs.ucdavis.edu).

2. Gathering results from *CoStmts*.

2.1. JR extensions and terminology. This paper uses only a small subset of the JR extensions to Java. In particular,

- JR provides threads[1] built on top of Java threads. Within the JR implementation, creating a JR thread includes creating a Java thread and setting up various structures used in implementing JR features such as quiescence detection and allowing invocations across physical machines (using RMI) [1, 2, 9].
- JR's *operation* abstraction generalises a method. An operation can be invoked in two ways and serviced in two ways, which provides flexibility in solving concurrent programming problems. However, this paper requires only the "method" style of operation invocation and servicing. Unlike a Java method invocation, a new JR thread is created to execute the method code while the invoking thread waits. (However, some optimisations allow the invoking thread to execute the method code directly [1, 2], but those optimisations do not apply for concurrent invocations from a *CoStmt*.)
- JR provides a concurrent invocation statement, which is introduced via examples in Sect. 2.2. However, this paper uses only a subset of the *CoStmt*.

TABLE 2.1
Declarations and other code used in the examples. (The declarations are used where needed; Copy-A and Init-L are used only when explicitly stated.)

Decl-A:	Declaration of array `A`.	`int [] A = new int[N];`

Decl-f:	Declaration of operation `f` and its code.	`private static op int f(int i) {`
		`  return i;`
		`}`

Decl-L:	Declaration of `ArrayList L`.	`ArrayList<Integer> L = new ArrayList<Integer>(N);`

Copy-A:	Copy array `A` to `ArrayList L`.	`for (int a: A) {`
		`  L.add(a);`
		`}`

Init-L:	Initialise `ArrayList L`.	`for (int i = 0; i < N; i++) {`
		`  L.add(null);`
		`}`

2.2. Example *CoStmts* using an array and their properties. A typical use of a *CoStmt* is as follows. Suppose we have an array as declared in Decl-A (Tab. 2.1) and want each of its `N` elements `A[i]` to be assigned the value of `f(i)`, where operation `f` is as shown in Decl-f (Tab. 2.1).

Co-1/AD in Tab. 2.2 shows a *CoStmt* that solves this problem. This *CoStmt* contains only one "arm" (as do all *CoStmts* in this paper). The arm contains a *quantifier* that indicates that the assignment to `A[i]` and invocation `f(i)` is to be done for the specified range of values of `i`. The thread executing a *CoStmt* (the "parent" thread) initiates all the specified invocations in parallel. A separate thread is created to evaluate each invocation of `f(i)`.[2] The parent thread then waits for the invocations to complete. When an invocation completes, the parent thread assigns the return value from the invocation to `A[i]`, i.e., the parent thread executes the assignments to `A[i]` one at a time. After all invocations have completed, the parent thread continues executing after the *CoStmt*.

[1] This paper uses the term "JR thread", although the other JR literature uses the traditional term "process" to represent a separate thread of control.

[2] In practice, each invocation of `f` should require sufficient computation to justify creating a new thread to evaluate it. Here, for simplicity of exposition, it does not.

TABLE 2.2

CoStmts *and wrappers used as examples. Our notation for referring to an example* CoStmt *includes a number indicating the order in which it appears in this table (or 0 if it is only discussed in the paper) and two letters: the first is 'A' for array, 'L' for* `ArrayList`*, 'S' for* `synchronizedList`*, or 'V' for* `Vector`*; the second letter indicates where the element is assigned: 'D' for directly in the assignment containing the invocation, 'P' for in the PPC, or 'W' for in the wrapper. An example wrapper is given the same number as its associated* CoStmt*.*

Co-1/AD:	`co (( int i = 0; i < N; i++ )) A[i] = f(i);`

Co-2/AP:	`int fi;`
	`co (( int i = 0; i < N; i++ )) fi = f(i) {`
	`  A[i] = fi;`
	`}`

Co-3/AW:	`co (( int i = 0; i < N; i++ )) invokeAndAssign( A, i );`

Wr-3:	`private static op void invokeAndAssign(int [] A, int i) {`
	`  A[i] = f(i);`
	`}`

Co-4/AD:	// same as Co-1/AD, but then copies `A` to `L` (Copy-A (Tab. 2.1))

Co-5/LP: (*incorrect*)	`int fi;`
	`co (( int i = 0; i < N; i++ )) fi = f(i){`
	`  L.add(fi);`
	`}`

Co-6/LW: (*incorrect*)	`co (( int i = 0; i < N; i++ )) wrapperAdd(L, f(i));`

Wr-6: (*incorrect*)	`private static op void wrapperAdd(ArrayList<Integer> L, int fi) {`
	`  L.add(fi);`
	`}`

Co-7/LW: (*incorrect*)	`co (( int i = 0; i < N; i++ )) invokeAndAdd(L, i);`

Wr-7: (*incorrect*)	`private static op void invokeAndAdd(ArrayList<Integer> L, int i) {`
	`  L.add(f(i));`
	`}`

Co-8/LP	// first initialise `L` (Init-L (Tab. 2.1)); then:
	`int fi;`
	`co (( int i = 0; i < N; i++ )) fi = f(i){`
	`  L.set(i, fi);`
	`}`

Co-9/LW:	// first initialise `L` (Init-L (Tab. 2.1)); then: `co (( int i = 0; i < N; i++ )) invokeAndSet(L, i);`

Wr-9:	`private static op void invokeAndSet(ArrayList<Integer> L, int i) {`
	`  L.set(i, f(i));`
	`}`

An arm of a *CoStmt* can also specify *post-processing code (PPC)*. The parent thread executes the PPC as each invocation completes; the arm's quantifier variables are accessible within the PPC. Thus, for example, Co-1/AD can be written equivalently using a PPC as in Co-2/AP, again insuring that the assignments to `A[i]`

are executed one at a time.

Each of Co-1/AD and Co-2/AP have the properties:

PropParallel: The invocations `f(i)` are evaluated (at least conceptually) in parallel.

PropSynch: The assignments to `A[i]` are synchronised, i.e., only the parent thread accesses `A` so no data races occur.

PropOrder: The assignments to `A[i]` give the desired order i.e., on completion, `A[i]` contains `f(i)`. Note, though, that assignments to `A` are *not* guaranteed to occur in the order of `i` (0, 1, 2, ...) due to the nondeterministic ordering of execution of the threads that evaluate `f(i)`.

Co-3/AW shows another *CoStmt*, which uses the wrapper Wr-3. It is like Co-1/AD and Co-2/AP in that it has the PropParallel, PropOrder, and PropSynch properties. However, by placing the assignment to `A[i]` within `invokeAndAssign`, it permits parallel assignments to different elements of `A` (whereas recall that Co-1/AD and Co-2/AP serialised these assignments).

2.3. Using an `ArrayList` instead of an array. We now consider a problem similar to that in Sect. 2.2. Suppose we have an `ArrayList` [10] as declared in Decl-L (Tab. 2.1). We want to write a *CoStmt* to assign it values as we did for the array in Sect. 2.2, i.e., element 0 contains `f(0)`, etc.

Of course, we could use Co-1/AD or Co-2/AP, and then copy from the array into the `ArrayList` (Copy-A (Tab. 2.1)), i.e., Co-4/AD.

However, some might find using an array only so it can later be assigned to an `ArrayList` to be aesthetically unpleasant and might be concerned about potential extra costs in doing so (discussed in Sect. 3). So, the problem becomes:

> Can we directly use an `ArrayList` within the *CoStmt*?

2.3.1. Incorrect `ArrayList` solutions. Our first attempt, Co-5/LP, is similar to Co-2/AP: it uses the PPC for adding `fi` to `L`.

Co-5/LP, however, is *not* equivalent to Co-2/AP as it does not provide PropOrder: the invocations of `f(i)` can complete in any order, so their values can be added to `L` in any order, not necessarily in order of `i`. We saw that behaviour in our experiments (Sect. 3): in some executions, some elements of `L` were not in the correct places.

Alternatively, we can use a *CoStmt* that assigns directly to the `ArrayList`, similar in style to Co-1/AD, such as Co-6/LW. Co-6/LW uses a wrapper operation, `wrapperAdd` (Wr-6), since what the *CoStmt* can invoke must be a JR operation and `ArrayList`'s `add()` is just a regular Java method.

However, Co-6/LW, is *not* equivalent to Co-1/AD or Co-2/AP as it does not provide PropParallel. Since `f(i)` appears as a parameter of the invocations of `wrapperAdd()`, they are evaluated serially, by the parent thread, before the invocations occur. (Co-6/LW also has other problems, similar to those presented for other *CoStmts* in the following discussion.)

The simple fix to Co-6/LW is to change the code to "wrap" not only the `add()` but also `f(i)`, e.g., Co-7/LW and Wr-7. This code now provides PropParallel. However, it does not provide PropOrder for the same reason as given for Co-5/LP (and with the same behaviour observed in experiments). Moreover, worse than Co-5/LP, `add()` in `wrapperAdd()` is not properly synchronised so invocations of `add()` that occur in parallel are not guaranteed to work, as per Fig. 2.1. In our experiments, the effect was that some elements were "lost": the overall size of the `ArrayList` was less than `N`, i.e., it was missing some elements.[3]

> "Note that this implementation is not synchronized. If multiple threads access an ArrayList instance concurrently, and at least one of the threads modifies the list structurally, it must be synchronized externally. (A structural modification is any operation that adds or deletes one or more elements, or explicitly resizes the backing array; *merely setting the value of an element is not a structural modification* [emphasis added].)"

FIG. 2.1. *Excerpt from the Java API for* `ArrayList` *[10].*

[3]That behaviour is what we observed with the Java implementation we used; such behaviour is *not* guaranteed by the Java API for `ArrayList` [10], e.g., program execution might lead to other results or a runtime exception.

To have Co-7/LW provide PropSynch, we can change from using just an `ArrayList` to use a Java Collection that does provide synchronisation. `synchronizedList` [11], e.g.,

```
List<Integer> L = Collections.synchronizedList(new ArrayList<Integer>(N));
```

or a `Vector` [12], e.g.,

```
Vector<Integer> L = new Vector<Integer>(N);
```

As now expected, our experiments show that, in either case, `L` has all of the desired `N` elements.

However, neither change provides PropOrder.

2.3.2. Correct `ArrayList` solutions. To have the code provide PropOrder, we take note of the emphasised part of Fig. 2.1. Co-8/LP now initialises the `ArrayList` so that all its elements are `null` (Init-L (Tab. 2.1)). That allows the PPC to use `set()` to place the element at the correct location in `L`.

This solution requires that `L` be just an `ArrayList`, i.e., it does *not* require `L` to be a `synchronizedList` or a `Vector` (although it would also work with either of those). The code is now not using any `ArrayList` method that is making a "structural modification"; it is just setting element values.

Symmetric to Co-3/AW, we can also use a wrapper, `invokeAndSet`, that both invokes first `f` and then `set()`, as seen in Co-9/LW and Wr-9.

Note how the correct `ArrayList` solutions require an exact bound (or at least an upper bound) on `N`. Such a bound is required, for example, so that the code has an `i` with which to `set(i, f(i))`. Needing `i` here is similar to needing an identifying index along with partial answers in the reply messages in other applications, such as pipeline sort [2].

3. Performance. This section presents performance comparisons in terms of execution times. The reported times are for only the critical part of program execution: the execution of the *CoStmt* (and any initialisation), not program startup or input/output.

3.1. Platforms. The data presented were obtained on a 4-core 4.00GHz system with 16 gigabytes of RAM running Fedora Core 24 Linux (4.10.10-100.fc24.x86_64). We ran the tests when the system was lightly-loaded. The data are based on the averages over several executions; variances were not significant. We also ran the tests on two other multicore platforms: a 3.60GHz system with otherwise identical hardware and software as our just-mentioned test system; and a 4-core 2.80GHz system with 16 gigabytes of RAM running Ubuntu 16.04 Linux (4.4.0-75-generic). The specific results differed, but they showed the same general trends. The specific versions of software that we used are JR 2.00607 [5] built with Java 1.8.0_60.

3.2. Test programs. The programs we tested were based on the *CoStmts* in Tab. 2.2. Specifically, we report on only the performance of the *correct* solutions. However, we observed that when the incorrect solutions ran properly their execution times were also close to the executions times reported for the correct solutions.

Table 3.1 and Fig. 3.1 present the data for those *CoStmts*. The data show little differences in the performances. The dominant cost is the `N` invocations from the *CoStmt*. Each such invocation creates a new JR thread, which is expensive: it includes creating a Java thread, plus additional overhead for implementing the additional JR features, e.g., those noted in Sect. 2.1. The work done by each thread is minimal, e.g., evaluating `f(i)`, whose computation is trivial.

The differences in whether the code uses an array, `ArrayList`, `synchronizedList`, or `Vector`, or uses PPC, wrappers, or neither makes no significant difference in performance in these *CoStmts*. However, one might prefer a particular *CoStmt* in Tab. 2.2 for stylistic reasons.

Similarly, in a more realistic program, where the cost of evaluating `f(i)` would be more than the cost of starting a thread, that cost would dominate the cost of potentially extra synchronisation.

3.3. Additional Java `List` tests. To compare the extra synchronisation costs of using Java's `synchronizedList` or `Vector` versus the non-synchronised `ArrayList`, we performed separate tests on a multithreaded Java program. To focus on the cost of synchronisation on the various `List` structures, versus the cost of thread creation, we limited the number of threads. This program creates a list with `E` elements and

TABLE 3.1
Execution times (in milliseconds) on the CoStmts *in Tab. 2.2 over a range of number of invocations (*`N`*).*

Number of invocations (`N`)	Co-1/AD	Co-2/AP	Co-3/AW Wr-3	Co-4/AD	Co-8/LP	Co-9/LW Wr-9	Co-0/SW Wr-0	Co-0/VW Wr-0
1	61	61	64	63	59	61	60	60
2	62	61	62	63	63	62	61	62
10	61	60	61	64	61	60	61	61
20	62	62	63	65	62	62	65	63
100	93	93	96	95	93	95	98	93
200	138	137	139	137	137	134	146	137
1000	906	908	906	909	905	903	858	911
2000	4534	4543	4570	4536	4551	4556	4543	4548

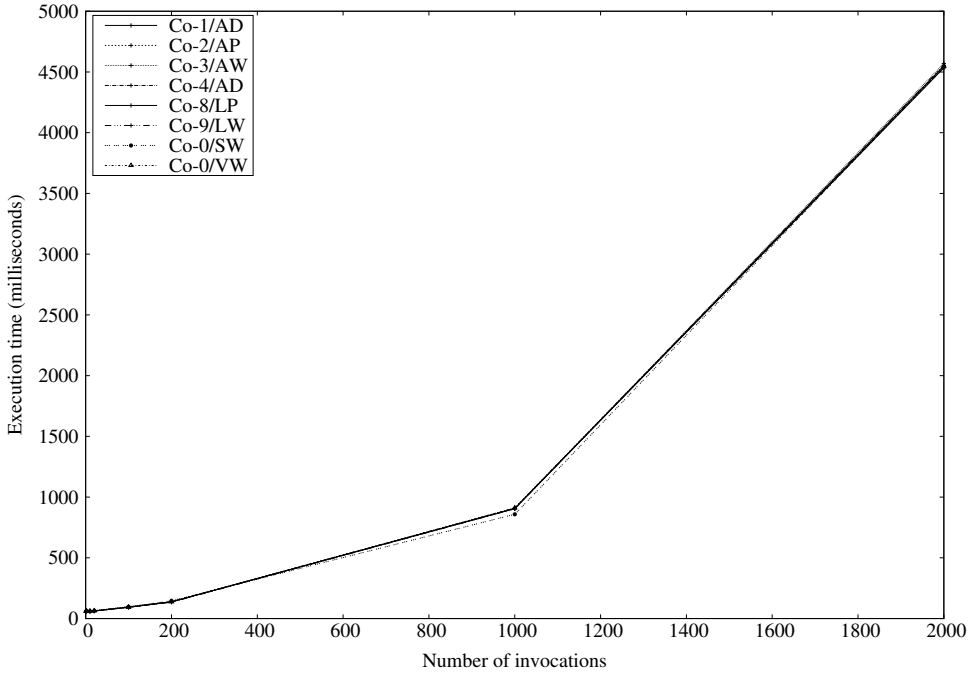


FIG. 3.1. *Execution times (in milliseconds) on the* CoStmts *in Tab. 2.2 over a range of number of invocations (*`N`*).*

then assigns values to each of the elements, much like the JR programs in Tab. 2.2. The program creates three threads that split the work: each thread is responsible for assigning values to (roughly) `E/3` elements in the list.

We tested two versions of this program for each kind of list: one version uses `add()` (similar to Co-7/LW) the other version uses `set()` (similar to Co-9/LW). The tests of the first version again showed that these solutions are incorrect for the same reasons given in Sect. 2.3.1. The tests of the second version confirmed it is correct and produced the data shown in Tab. 3.2 and Fig. 3.2. Note that these data are given in nanoseconds whereas the data in Tab. 3.1 and Fig. 3.1 are given in milliseconds.

The data show little differences in the performances of the three kinds of lists for the smaller values of `E`, i.e., less than 500 elements. However, after that point, the `ArrayList` version executes noticeably faster than the `synchronizedList` or `Vector` versions, presumably due to the extra cost of synchronisation. Also, after that point the `Vector` version executes a bit faster than the `synchronizedList` version.

The performance of these Java programs confirms the results discussed in Sect. 3.2 and seen in Fig. 3.1 and Tab. 3.1 for the JR programs. In particular, the relatively small differences in the costs of using an `ArrayList`, `synchronizedList`, or `Vector` are not noticeable within the JR programs.

TABLE 3.2
Execution times (in nanoseconds) of multithreaded Java `List` *programs over a range of number of elements (*`E`*).*

Number of elements (`E`)	`ArrayList`	`synchronizedList`	`Vector`
10	285409	300594	302091
50	304258	315736	305113
100	346149	355863	332241
500	543540	663671	548225
1000	819318	1185960	855099
5000	1632873	2566195	2025190
10000	2251727	4088558	3368276
50000	4786085	8395234	8381863
100000	6454248	13450121	12783344
500000	19583036	39482694	39359887

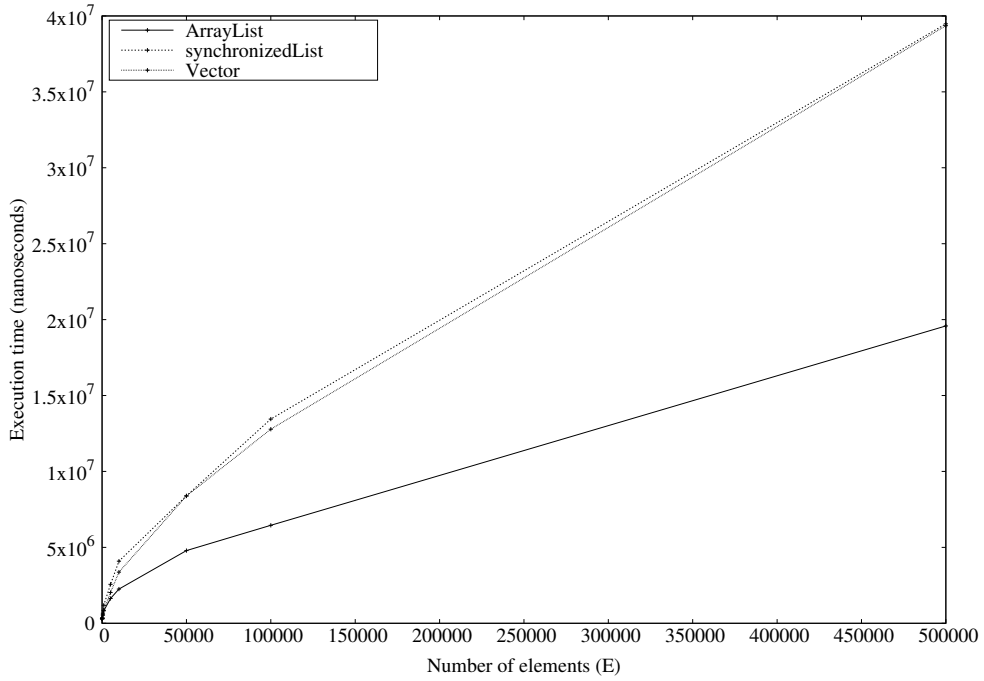


FIG. 3.2. *Execution times (in nanoseconds) of multithreaded Java* `List` *programs over a range of number of elements (*`E`*).*

4. Conclusion. This paper has presented a problem that arises in using Java `List` structures in gathering results from the *CoStmt*. It presented incorrect and correct attempts at solutions and discussed the tradeoffs among the correct solutions. Some of these solutions require less synchronisation. This paper also examined the performance of the correct solutions and further explored the costs of synchronisation in the underlying Java `List` structures.

The results show that, for this problem, the performances of the various correct *CoStmts* are roughly the same. Despite some requiring extra synchronisation, that extra cost is not significant compared to the larger costs inherent in the program.

The results comparing the costs of synchronisation in the underlying Java `List` structures and using initialisation and `set()` to provide the needed order can also be applied to regular Java threads programming and perhaps to programming in other concurrent languages and systems with similar features.

Acknowledgments. Nancy J. Wilson gave helpful feedback on the content and presentation of this paper. The anonymous reviewers carefully read this paper and gave helpful comments (and very promptly), which helped us improve this paper. The Editor-in-chief, Prof. Dana Petcu, responded promptly and encouragingly

to our queries.

REFERENCES

[1] A. W. Keen, T. Ge, J. T. Maris, and R. A. Olsson. JR: Flexible distributed programming in an extended Java. *ACM Transactions on Programming Languages and Systems*, pages 578–608, May 2004.

[2] R. A. Olsson and A. W. Keen. *The JR Programming Language: Concurrent Programming in an Extended Java*. Kluwer Academic Publishers, Inc., 2004. ISBN 1-4020-8085-9.

[3] G. R. Andrews, R. A. Olsson, M. Coffin, I. Elshoff, K. Nilsen, T. Purdin, and G. Townsend. An overview of the SR language and implementation. *ACM Transactions on Programming Languages and Systems*, 10(1):51–86, January 1988.

[4] G. R. Andrews and R. A. Olsson. *The SR Programming Language: Concurrency in Practice*. The Benjamin/Cummings Publishing Co., Redwood City, CA, 1993.

[5] JR language implementation, last accessed 2015-04-04 . `http://www.cs.ucdavis.edu/~olsson/research/jr/`.

[6] H. N. (Angela) Chan, E. Pauli, B. Y. Man, A. W. Keen, and R. A. Olsson. An exception handling mechanism for the concurrent invocation statement. In Jose C. Cunha and Pedro D. Medeiros, editors, *Euro-Par 2005 Parallel Processing*, number 3648 in Lecture Notes in Computer Science, pages 699–709, Lisbon, Portugal, August 2005. Springer–Verlag.

[7] Message Passing Interface Forum, last accessed 2015-04-04. `http://www.mpi-forum.org/`.

[8] Programming with the .NET framework, last accessed 2015-04-04. `http://msdn.microsoft.com/en-us/library/aa720433(v=vs.71).aspx`.

[9] R. A. Olsson and T. Williamson. RJ: A Java package providing JR-like concurrent programming. *SOFTWARE–Practice and Experience*, 46(5):685–708, 2016.

[10] Class ArrayList<E>, last accessed 2017-04-15. `https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html`.

[11] Class Collections, last accessed 2017-04-15. `https://docs.oracle.com/javase/8/docs/api/java/util/Collections.html`.

[12] Class Vector<E>, last accessed 2017-04-15. `https://docs.oracle.com/javase/8/docs/api/java/util/Vector.html`.

Edited by: Dana Petcu
Received: May 2, 2017
Accepted: Jul 25, 2017

DOI 10.12694/scpe.v18i3.1306
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 18, Number 3, pp. 261–274. http://www.scpe.org

A BSPLIB-STYLE API FOR BULK SYNCHRONOUS PARALLEL ML

FRÉDÉRIC LOULERGUE*

Abstract. Bulk synchronous parallelism (BSP) offers an abstract and simple model of parallelism yet allows to take realistically into account the communication costs of parallel algorithms. BSP has been used in many application domains. BSPlib and its variants are programming libraries for the C language that support the BSP style.

Bulk Synchronous Parallel ML (BSML) is a library for BSP programming with the functional language OCaml. It offers parallel operations on a data structure named parallel vector. BSML provides a global view of programs, i.e. BSML programs can be seen as sequential programs working on a parallel data structure (seq of par) while a BSPlib program is written in the SPMD style and understood as a parallel composition of communicating sequential programs (par of seq). The communication styles of BSML and BSPlib are also quite different.

The contribution of this paper is a BSPlib-style communication API implemented on top of BSML. It has been designed without extending BSML, but only using the imperative features of the underlying functional language OCaml. Programs implemented using this API are syntactically very close to programs implemented using a BSPlib library for the C language. It therefore shows that BSML is universal for the BSP model.

Key words: bulk synchronous parallelism, parallel programming, functional programming

AMS subject classifications. 68N18, 68M19, 68W10

1. Introduction.

Introduced by Valiant [28] and McColl [20], bulk synchronous parallelism (BSP) is a model of parallelism that offers a high level of abstraction yet takes realistically into account communications and synchronization. Works on BSP algorithms are numerous and BSP has been used successfully for a broad variety of applications: scientific computation [2], artificial intelligence [5, 6, 25], parallel databases, *etc.* It has also inspired more recent frameworks such as Pregel [19], and open source frameworks such as Giraph[1].

The BSP model considers a BSP computer to be a distributed memory machine: p processor-memory pairs connected though a network allowing point-to-point communications, and a global synchronization unit. However this architecture should a thought as an abstraction one: The BSP model targets all general purpose parallel architectures. Such architectures can always be seen as BSP computers. For example shared memory machines as well as clusters of PC can be seen as BSP computers even if some of their components are implemented as software instead of being implemented as hardware.

A BSP program is a *sequence* of *super-steps*. The parallelism therefore resides in each super-step. A super-step is composed of three phases: A computation phase where each processor computes using only the data it holds in its local memory; a communication phase where processors request data from other processors; and finally a synchronization phase which ends the super-step. The data requests are guaranteed to have been fulfilled when the synchronization phase ends. This structured form of parallel programs is the basis for the BSP *cost model*. In this context, cost means the estimate of parallel execution time. For the sake of conciseness, we omit the description of the cost model.

BSP algorithms can of course be implemented using a general message passing programming library such as MPI. However it is possible to optimize communications by taking into account the specific structure of BSP programs. Therefore, soon after the BSP model was introduced, two main libraries to support BSP programming were developed: the Green BSP library [9], and the Oxford BSP tool set [22]. To promote the adoption of the BSP model, the developers of these libraries produced a standard proposal: The BSPlib library [11]. Several implementations of this standard exist, some proposing extensions to the standard [1, 3, 27, 29, 30]. All these libraries are for the C imperative programming language (or more recently C++), and follow an imperative programming style.

Bulk Synchronous Parallel ML or BSML [4, 16] is also a programming library that supports the BSP model, but for the functional programming language OCaml[2]. The style of this library is purely functional and there

*School of Informatics, Computing and Cyber Systems, Northern Arizona University, Flagstaff, USA, frederic.loulergue@nau.edu

[1] http://giraph.apache.org

[2] http://www.ocaml.org

even exists a formal semantics as an extension of the λ-calculus [17].

It has been often claimed that BSML is universal for BSP programming, meaning that any BSP algorithm can be implemented using BSML. However most of the published BSP algorithms are written in an imperative style and with imperative data structures. Even if OCaml supports the imperative, functional and object-oriented programming styles, communications and synchronizations styles are very different between BSPlib and BSML. It is thus interesting to explore that claim and check whether imperative BSP algorithms could be implemented using BSPlib-style programming in BSML.

The contribution of this paper demonstrates that indeed any imperative BSP algorithm can be implemented with BSML using only the BSML pure functional primitives and some imperative features of its host language OCaml.

The paper is organized as follows. In Section 2, we present BSP programming using BSPlib and using BSML and we emphasize the differences between these two styles of programming. Section 3 is devoted to a BSPlib-like programming API for BSML, from the point of view of the user of the API. Some examples are implemented with this API and compared with their BSPlib counterparts. We present the API from the point of view of the implementer in Section 4. This API is compared with a BSPlib implementation from a performance point of view in Section 5. Related work is discussed in Section 6. We conclude and give future research directions in Section 7.

This is an extended version of [15]: the current paper offers improved presentations of programming with BSPlib and BSML, of the API, as well as additional examples. Sections 4, 5, and 6 are completely new.

2. Bulk Synchronous Parallel Programming.

2.1. Programming with BSPlib. BSPlib is a standard proposal for C libraries supporting the BSP model. Some of the implementations of BSPlib provide extensions to the standard. We focus on the standard, and more precisely on the Direct Remote Memory Access (DRMA) communication primitives. The Message Passing sub-part of the standard could be treated in a similar way than the DRMA part. Moreover, they are more BSP algorithms implemented using the DRMA sub-part of the standard.

The BSPlib standard features initialization and finalization phases of a BSPlib program:

```
void bsp_begin(int maxprocs);
void bsp_end();
```

However these phases are implicit in BSML, so we do not describe them more here.

A BSPlib program follows the Single Program Multiple Data (SPMD) paradigm. BSPlib programmers write one program parametrized by the process identifier, and the parallel program is the (implicit) parallel composition of several instantiations of this parametrized communicating sequential program.

To access the processor identifier and the total number of processors, BSPlib provides respectively the following two functions:

```
int bsp_pid();
int bsp_nprocs();
```

Before being able to read/write from the memory of another processor, processors should indicate that some memory locations will be able to be accessed by distant processors. This is done using the following functions:

```
void bsp_push_reg(const void *ident, int size);
void bsp_pop_reg (const void *ident);
```

When a processor calls one of these two functions, all the processors should perform a call to the same function. The registration and un-registration are active only *after* the super-step where the call is done. It is invalid to try to un-register an unregistered address.

The current super-step ends when all the processors call the function:

```
void bsp_sync();
```

There are two main functions to read from/write to remote memory:

```
  void bsp_put(int pid, const void *src,
               void *dst, int offset, int nbytes);
  void bsp_get(int pid, const void *src,
               int offset, void *dst, int nbytes);
```

`bsp_put(pid, src, dst, offset, nbytes)` requests the writing of `nbytes` bytes of data from memory address `src`, of the process executing the function, to memory address `dst` (plus `offset` bytes displacement from `dst`) of processor `pid`. The destination address should have been previously registered.

`bsp_get(pid, src, offset, dst, nbytes)` requests the reading of `nbytes` bytes of data from address `src` (plus `offset`) of processor `pid` and writing into the address `dst` of the processor executing the call. The source address should have been previously registered.

The data is guaranteed to have been exchanged at the end of the super-step, i.e. after a call to `bsp_sync`.

```
void shift(int * value)
{
  int dst = (bsp_pid()+1)%bsp_nprocs();
  bsp_push_reg(value, sizeof(int));
  bsp_sync();
  bsp_put(dst , value, value, 0, sizeof(int));
  bsp_pop_reg(value);
  bsp_sync();
}

void scatter(int root, int * a, int * v)
{
  bsp_push_reg(v, sizeof(int));
  bsp_sync();
  if(bsp_pid()==root){
    for(int dst=0;dst<bsp_nprocs();dst++)
      bsp_put(dst,&a[dst],v,0,sizeof(int));
  }
  bsp_pop_reg(v);
  bsp_sync();
}

void gather(int root, int * value, int * array)
{
  bsp_push_reg(array,bsp_nprocs()*sizeof(int));
  bsp_sync();
  bsp_put(root, value, array, bsp_pid()*sizeof(int), sizeof(int));
  bsp_pop_reg(array);
  bsp_sync();
}
```

FIG. 2.1. *BSPlib for C Examples*

Figure 2.1 presents several examples of functions written using BSPlib. These are collective communication functions: all the processors are supposed to call the same function with coherent parameters. For example the `root` argument should be the same for all processor identifiers.

`shift` shifts "to the right" the content of an `int` variable on each processor. In a non SPMD view, `value` could be seen as a parallel array (of size `bsp_nprocs()`) of integers. `shifts` shifts circularly this array to the right.

`scatter` scatters the content of an array `a` of size `bsp_nprocs()` at processor `root` to an `int` variable `v` on each processor. This function must be called with the same value of `root` on all the processors and at processor `root` array `a` should contain at least `bsp_nprocs()` `int` values.

`gather` is the dual of `scatter`: It gathers all of each `int` value on each processor in an array in the memory of the `root` processor. `root` should have the same value on all the processors, while `array` only needs to be allocated and have size at least `bsp_nprocs()` on processor `root` only.

2.2. Programming with BSML. The remaining of the paper assumes some familiarity with a statically typed higher-order functional programming language such as Haskell, Standard ML or OCaml. A concise introduction to OCaml is [23].

Bulk Synchronous Parallel ML or BSML is an explicit parallel functional language, extension of the ML family of functional languages. Currently there exists an implementation as a library for the OCaml language, implemented on top of MPI. Compared to the implementation of a full BSML *language*, the current implementation does not provide a type checker specific to BSML code (even if such a type checker has been designed [8]). In essence the BSML programming style is purely functional. It is nevertheless possible to use the imperative features of OCaml in combination to BSML. However in this case, some care is required: some unsafe programs could be written (an implementation of the type checker previously mentioned would reject such programs).

BSML offers four constants to access the parameters of the underlying BSP architecture: **bsp_p** returns the number of processors in the parallel machine. The other parameters are omitted in this paper as they are related to the BSP cost model.

BSML parallelism is based on a parallel data structure named *parallel vector*. In a parallel vector, each processor contains one value of the vector. Using this data structure, BSML offers a global view of parallel programs, i.e. a program looks like a sequential program but operates on parallel data structures. It is very different from the SPMD paradigm: The global parallel structure of SPMD programs is much harder to understand than programs that offer a global view. Parallel vectors are the only way to obtain different values at different processors. All code outside parallel vectors is *replicated* among the processors and assured (if the program is purely functional) to be consistent everywhere. This view of replication is however the view of the BSML implementer: for the developer of BSML programs, code outside parallel vectors is just usual sequential code.

Parallel vectors have a polymorphic type: `'a par`, meaning a parallel vector where each processor contains a value of type `'a`. OCaml is a higher-order functional language: `'a` could be replaced by any type, including a type of function. There is one restriction: Nesting is forbidden so `'a` could not be a parallel type. The type system [8] also rejects programs with such nesting.

It is important to note that, like other high-level abstractions such as Powerlists [24], there is no index notation for parallel vectors. In other words, there is no direct access to individual values in parallel vectors: Vectors are handled globally through four BSML primitive functions.

The function **mkpar**: (int→ 'a) → 'a par builds a parallel vector from a function f. At processor i the vector will have value (f i). For example with 8 processors[3]:

```
# let r = mkpar (fun i->2*i);;
val r : int par = <0, 2, 4, 6, 8, 10, 12, 14>

# let l =
    let f i = (i-1) %% bsp_p in
    mkpar f;;
val l : int par = <7, 0, 1, 2, 3, 4, 5, 6>
```

where # is the prompt of the interactive loop, and the answer has the form *name : type = value* and, in the interactive loop, parallel vectors are written $\langle\, a_0\, , \ldots ,\, a_{p-1}\, \rangle$. %% is a modulo operator that always returns a positive number.

[3]We show here the evaluation of BSML expressions inside the BSML interactive loop.

OCaml is a higher-order language, functions are first class citizens. It is therefore possible to build a parallel vector of functions:

```
# let vf = mkpar (fun i -> fun x -> x + i);;
val vf : (int->int) par = < <fun>, ..., <fun> >
```

vf is a parallel vector where each processor i contains the function **fun** x → x + i. In the toplevel, the value representing a function is symbolized by an abstract value <fun>.

A parallel vector of functions is not a function: It cannot be applied to another parallel vector. A BSML primitive is needed to apply point-wisely a parallel vector of functions to a parallel vector of values. For example:

```
# let vv1 = apply vf r;;
val vv1 : int Bsml.par = <0, 3, 6, 9, 12, 15, 18, 21>
```

The type of **apply** is ('a→ 'b) par → 'a par → 'b par.

mkpar and **apply** only operate in the computation phase of a BSP super-step. Communications and implicit global synchronizations are performed using the primitives **proj** and **put**.

The function **proj**: 'a par → (int→ 'a) is the almost the dual of **mkpar**: from a parallel vector it creates a (replicated) function. However for a function f, **proj**(**mkpar** f) is different from f: f may be defined for a negative input, or for an input greater or equal to **bsp_p**, but **proj**(**mkpar** f) is only defined on the interval $[0, \mathbf{bsp\_p} - 1]$.

```
# let fr = proj r;;
val fr: int -> int = <fun>

# let four = fr 2;;
val four : int = 4
```

proj requires communications and a synchronization barrier to be evaluated. For more involved communication patterns, one needs to use the **put** function. It allows any local value to be transferred to any other processor. As **proj**, it ends the current super-step by a synchronization barrier.

The type if put is (int→ 'a)par→ (int→ 'a)par. A canonical use of **put** is **put** (**mkpar** (**fun** src dst → e)) where expression e computes (or usually, selects) the data that should be sent (depending on the processor identifier src) to processor dst.

Both the input and the result of the evaluation of a call to **put** are vectors of functions. In the input, each function encodes the messages to be sent by one processor to all the other processors, while in the output each function encodes the messages received by one processor from all the other processors. At a processor j the function obtained by a call to **put**, when applied to i, yields the value *received from* processor i by processor j.

For example, shifting the values of a parallel vector to the right could be written:

```
# let shift vv =
   let msg src v dst = if dst=(src+1) mod bsp_p then [v] else [] in
   let msgs = apply (mkpar msg) vv in
   let srcs = mkpar (fun i->(i-1)%%bsp_p) in
   parfun List.hd (apply (put msgs) srcs)
val shift : 'a par -> 'a par = <fun>

# let vv2 = shift (mkpar string_of_int);;
val vv2 : string par = <"7", "0", "1", "2", "3", "4", "5", "6">
```

where **let** parfun f v = **apply** (**mkpar**(**fun** i→ f)) v and List.hd returns the head of a non empty list.

In this example, the empty list [] is used as a way to mean "no message". The empty list is not sent to other processors. For any sum type the first constant constructor is considered to mean "no message". Even if **put** looks like a total exchange, it is not due to these special values.

The shift function proceeds as follows. msgs is a parallel vector of functions. At each processor src, it contains a function that returns the local value of vector vv (encapsulated in a singleton list) if applied to a process identifier equals to (src+1) **mod bsp_p** (the processor "to the right" of src, the processor to the right of

```
let shift (value: int ref) =
  begin
    bsp_push_reg value (ref int);
    bsp_sync();
    let dst = (bsp_pid()+1) mod bsp_nprocs() in
    bsp_put dst !value value int;
    bsp_pop_reg value (ref int);
    bsp_sync();
  end

let gather (root: int) (value: int) (a:int array) =
  begin
    bsp_push_reg a (array int);
    bsp_sync();
    bsp_put_sa root value a (bsp_pid()) int;
    bsp_pop_reg a (array int);
    bsp_sync()
  end

let scatter (root: int) (a: int array) (value: int ref) =
  begin
    bsp_push_reg value (ref int);
    bsp_sync();
    if (bsp_pid() = root) then
      begin
        for dst=0 to bsp_nprocs() − 1 do
          bsp_put dst a.(dst) value int;
        done
      end;
    bsp_pop_reg value (ref int);
    bsp_sync()
  end
```

FIG. 3.1. *BSPlib for BSML Examples*

the last processor being the first processor) and returns the empty list otherwise. Thus each of the functions of msgs encodes the messages to be sent by one processor to other processors, and some of the values mean "no message". (**put** msgs) is also a vector of functions. Now each of these functions encodes the messages received from other processors. We are only interested in messages coming from the processor immediately "to the left". The vector srcs contains at each processor the processor identifier of the processor to its left. Applying the vector of functions (**put** msg) to srcs therefore yields at each processor the message received from its left neighbour. As the value has been encapsulated into a singleton list, we need to apply, at each processor, the function List.hd.

3. An API for Imperative BSP Programming in BSML. In this section we present how to program with the proposed BSPlib API for OCaml implemented on top of BSML. The examples of BSPlib for C presented in Figure 2.1 are translated to this new API for OCaml in Figure 3.1. The imperative BSP programming styles for both C and OCaml are very close. With BSML it is therefore possible to use both styles: the BSPlib-like imperative style and the original BSML pure functional style.

One main difference between OCaml and C is that OCaml is strongly typed. Therefore it is not possible to directly translate in OCaml the BSPlib for C primitives where all data is considered as arrays of bytes.

Moreover, by default all variables are immutable in OCaml, but variables which types are *references* or *arrays* (and some others omitted here).

Furthermore programming in BSML does not follow the SPMD paradigm but the global view paradigm. However the proposed API does follow the SPMD paradigm. We explain in Section 4 how we dealt with this problem.

Finally, the implementation of a BSPlib-like API in OCaml requires to store references in a table. This is not easy as a reference to an int has a different type than a reference to a float and that data structures in OCaml are usually parametric polymorphic, meaning a structure should contain values of the same type.

The solution to this problem is to use generalized abstract data types (GADT) [7] to have values representing types so that a value v and a value v_τ representing the type τ of the value v can be packed together into a value v_{packed} whose type does not depend on the type τ. We will discuss this technical point further in Section 4.

Memory visibility. The impact on the API is that instead of having the size of an address we want to register, we have an expression that represents the type of the value we want to register (or un-register):

```
bsp_push_reg: 'a → 'a Type.ty → unit
bsp_pop_reg: 'a → 'a Type.ty → unit
```

These two functions are polymorphic, but if they are called on values that are not mutable, an error occurs (unit is the type of the value () usually returned by functions that only produce side effects).

For example, **let** x = **ref** 0 **in** bsp_push_reg x int will register the reference x (to an int) for DRMA communications. **ref** in OCaml is applied to a value v and creates a reference to a value of the type of v, with initial value v. In the example, the content referenced by x is therefore mutable. It is mandatory to specify the type also for bsp_pop_reg while the C version only needs the address of the memory location to un-register.

Synchronization barrier. As in BSPlib for C, a call to bsp_sync() ends the super-step by a synchronization barrier. A call to bsp_pid() returns the processor identifier, and a call to bsp_nprocs() return the number of processors of the BSP machine.

DRMA communications. The type of function bsp_put is:

```
bsp_put: int → 'a → 'a ref → 'a Type.ty → unit
```

bsp_put pid v r ty writes the value v to (registered) reference r of processor pid. The type of value v should be represented by ty. With bsp_put, values can only be written as a whole, even if they are arrays. This is therefore less flexible that the corresponding BSPlib for C function where arrays can be considered partly using the offset and nbytes arguments.

To provide such a flexibility, and due to typing constraints, we propose two variants of bsp_put:

```
bsp_put_sa : int → 'a → 'a array → int → 'a Type.ty → unit
bsp_put_aa : int → 'a array → 'a array → int → int → 'a Type.ty → unit
```

bsp_put_sa is used to put a scalar value into a remote array: bsp_put_sa pid v a k ty writes value v whose type is represented by value ty at index k of the remote array a at processor pid. The type of a should be represented by the value array ty.

bsp_put_aa is used to deal with arrays both at the source and at the destination. bsp_put_aa pid a1 a2 offset size ty writes size values of array a1 whose *elements'* type is represented by value ty at offset offset of the remote array a2 at processor pid.

The imperative API also offers functions to read from remote memory:

```
bsp_get : int → 'a ref → 'a ref → 'a Type.ty → unit
```

and the associated _sa and _aa variants.

4. Implementation of the Imperative API. In the development of the imperative API, two difficulties had to be overcome: typing issues and offering a SPMD view for the API while BSML offers a global view. Before explaining how we dealt with these issues, let us describe briefly the data structures used to implement the API.

Data Structures. **bsp_put** and **bsp_get** can only target memory locations that have been previously registered. The first data structure is a table of registered memory locations, implemented as a mutable list of such memory locations (or references). As in OCaml, references to a value of type int has a different type than a reference to a value of type float, this list should contain values of the type Dyn.t described below.

In our API the message requests are actually exchanged when **bsp_sync** is called. Therefore we need a data structure to contain the message requests added to this structure by calls to **bsp_put** and **bsp_get**. This data structure is a mutable list of value of type request. This type has six constructors that correspond to the three versions of **bsp_put** and the three versions of **bsp_get** (The arguments to these six constructors are the arguments to these six functions but the destination reference which is replaced by the *position* of the reference in the registered memory location table).

Typing Issues. We use a recently introduced feature of OCaml: generalised algebraic data-types or GADT. GADT are extensions of sum types (or variant types). A sum type in OCaml is similar to a union type in C or a record type with variant parts in Ada. However there is no discriminant field. Instead each case is associated with a symbol, called constructor. Constructors are used to discriminate between values of the sum type. For example abstract syntax trees for a small language of integers and boolean expressions could be implemented as the following type:

```
type expr =
| Int of int
| Bool of bool
| Add of expr * expr
| And of expr * expr
```

It is of course possible to write meaningless (non well typed) expressions such as And(Int 0, Bool **true**). To reject such expressions it would be necessary to write a type checking function, for example:

```
type ty = | TBool | TInt
let rec type_of : expr → ty option = function
  | Int _ → Some TInt
  | Bool _ → Some TBool
  | Add(e1, e2) →
      if (type_of e1) = Some TInt && (type_of e2) = Some TInt
      then Some TInt else None
  | And(e1, e2) →
      if (type_of e1) = Some TBool && (type_of e2) = Some TBool
      then Some TInt else None
```

where the type 'a option is a sum type with too constructor: Some that takes as argument a value of type 'a and None.

type_of can therefore return a value of type ty embedded in a Some constructor if the input expression is well typed, and returns None if the expression is not well typed.

GADT introduce two novelties with respect to sum types: The possibility to have more constrained type parameters depending on the constructor, and the possibility to introduce existential type variables, i.e. using a type variable in a constructor case that is not one of the type parameters.

Using GADT it is possible to rewrite the previous example in such a way that it is not possible to write meaningless expressions, using the type system of OCaml:

```
type _ expr =
| Int: int → int expr
| Bool: bool → bool expr
| Add: int expr * int expr → int expr
| And: bool expr * bool expr → bool expr
```

Now the type expr has a type parameter (here written _ because we don't need it to have an explicit name) that allows to indicate the type of the expression. This allows to specify the type of expressions the constructors Add

and And take as argument. Trying to use And(Int 0, Bool **true**) would raise a compile time type error indicating that Int 0 has type int expr but that And expects a value of type bool expr.

It is possible to write a GADT such that values of this type *represent* the type of the GADT type parameter:

```
type _ ty =
| Int : int ty
(* ... *)
| Ref : 'a ty → 'a ref ty
| Array : 'a ty → 'a array ty
```

For example the value Array Int has type int array ty and represents the type int array. In order to ease the use of this GADT, we defined constants and functions to build values of type ty that are very close syntactically to the type they represents:

```
let int = Int
(* ... *)
let array a = Array a
```

For example the type int list array of arrays containing a list of integers in each cell is represented by the value array (list int). Note that function **ref** that builds a reference from a value is masked by our definition of **ref**. The former is now written P.**ref**.

To be able to store different communication requests into the same data structure, we need a way to store data in a type such that the type does not depend on the type of the stored data, but still allowing some type checking by OCaml, to avoid to write or read data of a wrong type to/from remote memory. To do so we use the second feature of GADT: existential types.

The module Dyn provides an abstract type t implemented as

```
type t = D: 'a ty * 'a → t
```

A value of type Dyn.t contains a value v of some type τ plus a value v_τ representing the type τ i.e. a value of type τ ty. Dyn also provides a function make: 'a ty → 'a → t to build values of type Dyn.t and update functions: update_ref, update_array and update_array_a. These functions take two values of type Dyn.t (and possibly an offset and a size) and update the first argument if it is a mutable value, with the value of the second argument, if their types are equivalent.

When all processors call bsp_push_reg r ty, the API stores make r ty of type Dyn.t in the table of registered memory locations. As all processors are supposed to call this function, there is a consistent indexing of visible memory locations.

When a processor calls bsp_put pid v r ty, the API stores make v ty of type Dyn.t in a table of communication requests together with the index of r in the table of registered memory locations. At a call to bsp_sync the requests are sent to the destination processors. As the data part of write/read requests are communicated as values of type Dyn.t, at destination the memory is updated using the update functions of module Dyn. When all the updates are done, the next super-step can begin.

Global view vs. SPMD. As a BSML program deals with a whole parallel machine and individual processors at the same time, a distinction between the levels of execution that take place is needed. *Replicated* execution is the default. Code that does not involve BSML primitives is run by the parallel machine as it would be by a single processor. Replicated code is executed at the same time by every processor, and leads to the same result everywhere. *Local* execution is what happens inside parallel vectors, on each of their components: The processor uses its local data to do computation that may be different from other processors. *Global* execution concerns the set of all processors together, but as a whole and not as a single processor, for example the use of communication primitives.

Consider the following program:

```
let x = ref 0 in
  let vv3 = mkpar(fun pid → x := pid) in
  if !x <> 0 then shift vv1 else vv1
```

The mutable variable x is created at the replicated level. Therefore its value would be the same on all the processors. However it is mutated differently on each processor at the local level inside **mkpar(fun** pid → x:=pid). The content of x is no longer replicated. This could cause a problem if the value of x is used again at the replicated or global level. In the example all the processors will call shift but processor 0. As synchronization barriers should be called by all the processors in BSP, this program fails.

Such an example is rejected by the type system [8]. However BSML current implementation is a library only: it does not come with a type checker. In the implementation of the API we use a replicate reference to store the process identifier and modify it locally so that the content of this reference is the processor identifier. That allows to implement bsp_pid and write programs in the SPMD style. Registered memory locations as well as the communication requests are also in replicated mutable variables but don't remain replicated due to the use of bsp_pid in the control flow. Note that the problem exposed above is still present, and it is actually quite easy to write incorrect BSPlib C programs from the point of view of synchronization.

However instead of using the type checker [8] it may be possible to relax the type system to allow the implementation of the imperative API to type check and add an adaptation of the static analysis proposed by Jakobsson et al. [12] to avoid synchronization errors.

5. Experiments. The current implementation of the proposed API is a proof of concept, and we favor simplicity and conciseness over performances. However, we experimented the performances of the BSML BSPlib API with respect to a BSPlib implementation in C, and a functional style BSML implementation. The experiments were conducted on a shared memory machine running Ubuntu Linux 16.04, with two Intel Xeon E5-2683 v4 processors (16 cores each) at 2.10 GHz and 256Gb of memory. The following libraries and compilers were used:

- BSPonMPI version 0.4.2,
- BSML version 0.5.4,
- OpenMPI version 1.5.4,
- OCaml version 4.04.0,
- GCC version 5.4.0 (with optimization flag -O3).

The test application is an inner product. The main function of the BSPlib version in C is shown in Figure 5.1 (taken from BSPEdupack [2]). p is the total number of processors, s is the local processor identifier, and n is the size of the vectors. The corresponding version using the proposed API for BSML is depicted in Figure 5.2. Basically only the local variable declarations are different. We also tested variant where bsp_sync() was replaced by Put.bsp_sync() a slightly optimized version that avoids to do a second "empty" synchronization barrier when no communication request is a bsp_get. Finally we tested a version mostly in the functional BSML style (Figure 5.3).

The experiments were done using respectively 2, 4, 8, 16 and 32 cores, and for two global size of vectors: 10^3 and 10^9. The results are presented in Figures 5.4 and 5.5: each figure indicates the median value of 100 measures as well as the standard deviation.

For size 10^3 the communication and synchronization costs are dominant and therefore the running time increases with the number of cores. BSML communication and synchronization phases are more expensive than BSPlib for C communication and synchronisation phases. The first reason is that the communication functions **proj** and **put** are polymorphic: serialization of the data to be exchanged is performed, and the resulting raw data is bigger than its non serialized counterpart. The second reason is that for **put**, the encapsulation of messages into functions makes necessary the application of each of these functions to all the process identifiers. OCaml sequential computations on arrays are also slightly less efficient than C sequential computations on arrays.

The BSPlib API for BSML is even less efficient: Indeed a call to **bsp_sync** in this case requires two underlying calls to **put**, but for the BSPlib+BSML Opt. version: its performance is closer to the functional BSML version. For the general bsp_sync() is because to implement **bsp_get** messages, a first call to put is needed to send a request for the data, and a second call to put is needed for the requested processors to send back the requested data. When a super-step contains only **bsp_put** messages, we optimized the implementation of the API to perform only one call to **put**. For size 10^9, all the versions based on BSML have very close performances.

The difference of performance for size 10^9 between the C version and the BSML versions is mostly due to the difference in sequential performance, and the serialization for data that both takes time and makes messages

```
double bspip(int p, int s, int n, double *x, double *y){
    Inprod= vecallocd(p); bsp_push_reg(Inprod,p*SZDBL);
    bsp_sync();

    inprod= 0.0;
    for (i=0; i<nloc(p,s,n); i++) inprod += x[i]*y[i];
    for (t=0; t<p; t++) bsp_put(t,&inprod,Inprod,s*SZDBL,SZDBL);
    bsp_sync();

    alpha= 0.0;
    for (t=0; t<p; t++) alpha += Inprod[t];
    bsp_pop_reg(Inprod); vecfreed(Inprod);

    return alpha;
}
```

FIG. 5.1. *Inner Product: BSPlib C Version*

```
let bspip (p:int) (s:int) (n:int) (x:float array) (y:float array) : float =
  let inprod_array = Array.make (bsp_nprocs()) 0.0 in
  let inprod = P.ref 0.0 and alpha = P.ref 0.0 in
  begin
    bsp_push_reg inprod_array (array float);
    bsp_sync ();

    for i=0 to (nloc p s n)−1 do inprod := !inprod +. x.(i) *. y.(i) done;
    for t=0 to p−1 do bsp_put_sa t !inprod inprod_array s float done;
    bsp_sync();

    for t=0 to p−1 do alpha := !alpha +. inprod_array.(t) done;
    bsp_pop_reg inprod_array (array float);

    !alpha
  end
```

FIG. 5.2. *Inner Product: BSPlib API for BSML Version*

bigger. The performance difference between the C version and the BSML functional version could be reduced in general, if BSML was extended with more specialized versions of **put** (and **proj**) so that the messages are not encapsulated into functions, and that for basic types, values are not serialized.

6. Related Work. Recent implementations of BSPlib for C include BSPonMPI [27], MultiCore BSP [30], and Zefiros BSP [29]. BSPonMPI targets distributed memory machines while MultiCore BSP and Zefiros BSP target shared memory machines. They are very close to the standard. There exists a BSPlib implementation for Java [10]. This implementation is closer to the C style libraries than a new design focusing on object orientation. The proposed API is therefore very close to all these libraries. The structured parallelism of BSP also allowed to design a "mock" BSPlib library for testing and debugging imperative BSP programs [26].

The Paderborn University BSPlib (PUB) [3] implements the standard but also adds subset synchronization to BSPlib (and the BSP model). Basically all the BSP primitives are given an additional first argument of type `tbsp`. This "BSP object" is very similar to an MPI communicator. It allows to create subgroups of processors, to ensure modularity, and to support different threads on the same processor running different BSP computations.

```
let ip (x: float array)(y:float array) =
  let inprod = P.ref 0.0 in
  for i=0 to (Array.length x)−1 do inprod := !inprod +. x.(i) ∗. y.(i) done;
  !inprod

let bspip (p:int) (s:int) (n:int) (x:float array par) (y:float array par) : float =
  begin
    let partial_ip = parfun2 ip x y in
    List.fold_left (+.) 0. (List.map (proj partial_ip) procs)
  end
```

FIG. 5.3. *Inner Product: BSML Version*

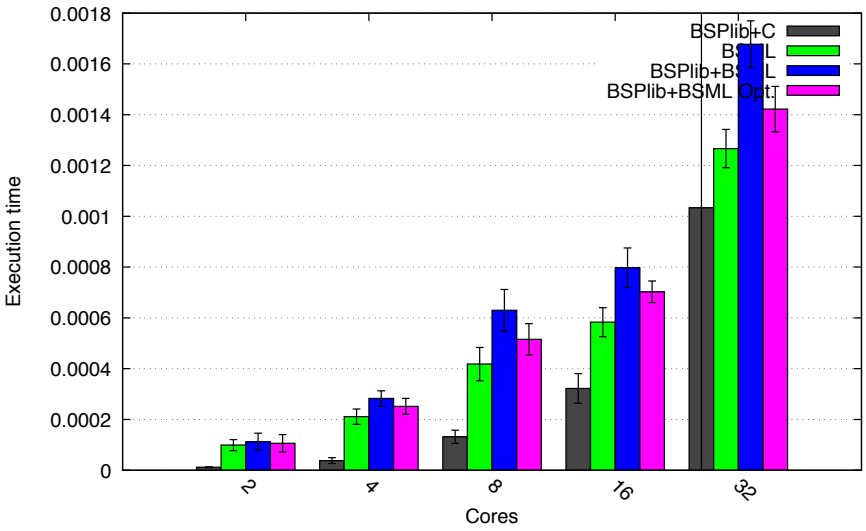


FIG. 5.4. *Experimental Results for Size* 10^3

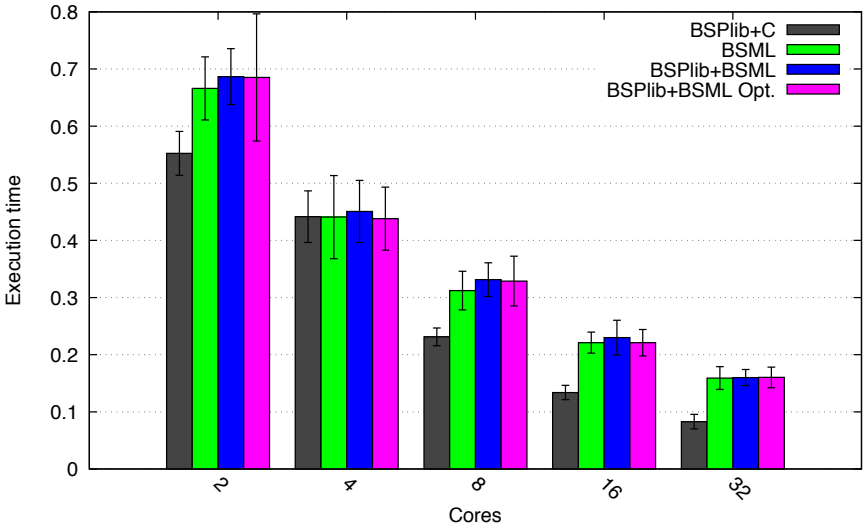


FIG. 5.5. *Experimental Results for Size* 10^9

Extensions of BSML feature two kinds of parallel compositions, juxtaposition [13] and superposition [14]. Juxtaposition supports subgroups of processors while still follows the pure BSP model: there is no subset synchronization. Superposition can be understood as a structured way to have several threads running different BSP computations. As it is structured, it is not as flexible as the PUB way of having multiple BSP threads. However this structure allows for safely *sharing* synchronization barriers between the threads rather than just ensuring that synchronization barriers do not interfere with each other. Therefore it would be possible to extend our API to handle some of the PUB extensions, subgroups and modularity, but with a different cost model.

There also exists a BSP programming library for the functional programming language Haskell [21]. The evaluation strategy of Haskell makes often more difficult to write BSP programs than BSML in OCaml. As Haskell is a pure functional language, a BSPlib-like API on top of [21] would be very difficult to design and not close to BSPlib C programming.

GADT are useful in the context of parallelism. We also used them for implementing recursive parallel data structures, namely Misra's powerlists [24], and associated operations in BSML while ensuring that no nesting of parallel vector occurs [18].

7. Conclusion and Future Work. In this paper we show that it is possible to implement a BSPlib-like imperative API using only Bulk Synchronous Parallel ML *purely functional* primitives together with the *sequential* imperative features of its host language OCaml. Bulk Synchronous Parallel ML is therefore universal for BSP programming.

This implementation relies on a recently introduced feature of OCaml: Generalized Algebraic Data Types (GADT). Experiments show reasonable performances for this proof-of-concept implementation of the API[4].

For full usability, the API should support also BSPlib message passing primitives. The type representation module we currently provide should also be extended in two ways: firstly it should support all the OCaml basic types, secondly it should be extensible in particular able to handle user-defined record types and sum types.

REFERENCES

[1] T. Bannink, A. Wits, and J.-W. Buurlage, *Epiphany BSP version 1.0*. http://github.com/coduin/epiphany-bsp, Jan. 2017.

[2] R. H. Bisseling, *Parallel Scientific Computation*, Oxford University Press, 2004.

[3] O. Bonorden, B. Judoiink, I. von Otte, and O. Rieping, *The Paderborn University BSP (PUB) library*, Parallel Computing, 29 (2003), pp. 187–207.

[4] W. Bousdira, F. Gava, L. Gesbert, F. Loulergue, and G. Petiot, *Functional Parallel Programming with Revised Bulk Synchronous Parallel ML*, in First International Conference on Networking and Computing (ICNC 2010), 2nd International Workshop on Parallel and Distributed Algorithms and Applications (PDAA), K. Nakano, ed., IEEE Computer Society, 2010, pp. 191–196.

[5] A. Braud and C. Vrain, *A parallel genetic algorithm based on the BSP model*, in Evolutionary Computation and Parallel Processing GECCO & AAAI Workshop, Orlando (Florida), USA, 1999.

[6] D. C. Dracopoulos and S. Kent, *Speeding up genetic programming: A parallel BSP implementation*, in First Annual Conference on Genetic Programming, MIT Press, July 1996.

[7] J. Garrigue and D. Rémy, *Ambivalent types for principal type inference with GADTs*, in Programming Languages and Systems, vol. 8301 of LNCS, Springer, 2013, pp. 257–272.

[8] F. Gava, L. Gesbert, and F. Loulergue, *Type System for a Safe Execution of Parallel Programs in BSML*, in 5th ACM SIGPLAN workshop on High-Level Parallel Programming and Applications, ACM, 2011, pp. 27–34.

[9] M. Goudreau, K. Lang, S. Rao, T. Suel, and T. Tsantilas, *Towards Efficiency and Portability: Programming with the BSP Model*, in Eighth Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA), New York, NY, USA, 1996, ACM, pp. 1–12.

[10] Y. Gu, B.-S. Lee, and W. Cai, *JBSP: A BSP programming library in Java*, Journal of Parallel and Distributed Computing, 61 (2001), pp. 1126–1142.

[11] J. M. D. Hill, B. McColl, D. C. Stefanescu, M. W. Goudreau, K. Lang, S. B. Rao, T. Suel, T. Tsantilas, and R. Bisseling, *BSPlib: The BSP Programming Library*, Parallel Computing, 24 (1998), pp. 1947–1980.

[12] A. Jakobsson, F. Dabrowski, W. Bousdira, F. Loulergue, and G. Hains, *Replicated Synchronization for Imperative BSP Programs*, in International Conference on Computational Science (ICCS), Procedia Computer Science, Zurich, Switzerland, 2017, Elsevier, pp. 535–544.

[13] F. Loulergue, *Parallel Juxtaposition for Bulk Synchronous Parallel ML*, in Euro-Par 2003, H. Kosch, L. Boszormányi, and H. Hellwagner, eds., no. 2790 in LNCS, Springer Verlag, 2003, pp. 781–788.

[4]Available at `http://traclifo.univ-orleans.fr/BSML`

[14] ———, *Parallel Superposition for Bulk Synchronous Parallel ML*, in International Conference on Computational Science (ICCS), no. 2659 in LNCS, Springer Verlag, 2003, pp. 223–232.
[15] F. Loulergue, *Imperative BSPlib-style Communications in Bulk Synchronous Parallel ML*, in International Conference on Computational Science (ICCS), Procedia Computer Science, Zurich, Switzerland, 2017, Elsevier, pp. 2368–2372.
[16] F. Loulergue, F. Gava, and D. Billiet, *Bulk Synchronous Parallel ML: Modular Implementation and Performance Prediction*, in International Conference on Computational Science (ICCS), vol. 3515 of LNCS, Springer, 2005, pp. 1046–1054.
[17] F. Loulergue, G. Hains, and C. Foisy, *A Calculus of Functional BSP Programs*, Sci Comput Program, 37 (2000), pp. 253–277.
[18] F. Loulergue, V. Niculescu, and J. Tesson, *Implementing powerlists with Bulk Synchronous Parallel ML*, in Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), Timisoara, Romania, 2014, IEEE, pp. 325–332.
[19] G. Malewicz, M. H. Austern, A. J. Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski, *Pregel: a system for large-scale graph processing*, in SIGMOD, ACM, 2010, pp. 135–146.
[20] W. F. McColl, *Scalability, portability and predictability: The BSP approach to parallel programming*, Future Generation Computer Systems, 12 (1996), pp. 265–272.
[21] Q. Miller, *BSP in a Lazy Functional Context*, in Trends in Functional Programming, vol. 3, Intellect Books, may 2002.
[22] R. Miller and J. Reed, *The Oxford BSP Library Users' Guide*, Oxford Parallel, Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford, OX1 3QD, 1.0 ed., May 1994.
[23] Y. Minsky, *OCaml for the masses*, Commun. ACM, 54 (2011), pp. 53–58.
[24] J. Misra, *Powerlist: A structure for parallel recursion*, ACM Trans Program Lang Syst, 16 (1994), pp. 1737–1767.
[25] R. O. Rogers and D. B. Skillicorn, *Using the BSP cost model to optimise parallel neural network training*, Future Generation Computer Systems, 14 (1998), pp. 409–424.
[26] W. Suijlen, *Mock BSPlib for Testing and Debugging Bulk Synchronous Parallel Software*, Parallel Processing Letters, 27 (2017), pp. 1–18.
[27] W. Suijlen and P. Krusche, *BSPonMPI*. https://github.com/pkrusche/bsponmpi, Apr. 2013. version 0.4.2.
[28] L. G. Valiant, *A bridging model for parallel computation*, Commun. ACM, 33 (1990), p. 103.
[29] M. van Duijn, K. Visscher, and P. Visscher, *BSPLib: a fast, and easy to use C++ implementation of the Bulk Synchronous Parallel (BSP) threading model.* `http://bsplib.eu/`, Sept. 2016. version 1.1.4.
[30] A. Yzelman, R. Bisseling, D. Roose, and K. Meerbergen, *MulticoreBSP for C: A High-Performance Library for Shared-Memory Parallel Programming*, International Journal of Parallel Programming, (2013), pp. 1–24.

Edited by: Dana Petcu
Received: Jul 7, 2017
Accepted: Sep 5, 2017

AIMS AND SCOPE

The area of scalable computing has matured and reached a point where new issues and trends require a professional forum. SCPE will provide this avenue by publishing original refereed papers that address the present as well as the future of parallel and distributed computing. The journal will focus on algorithm development, implementation and execution on real-world parallel architectures, and application of parallel and distributed computing to the solution of real-life problems. Of particular interest are:

Expressiveness:

- high level languages,
- object oriented techniques,
- compiler technology for parallel computing,
- implementation techniques and their efficiency.

System engineering:

- programming environments,
- debugging tools,
- software libraries.

Performance:

- performance measurement: metrics, evaluation, visualization,
- performance improvement: resource allocation and scheduling, I/O, network throughput.

Applications:

- database,
- control systems,
- embedded systems,
- fault tolerance,
- industrial and business,
- real-time,
- scientific computing,
- visualization.

Future:

- limitations of current approaches,
- engineering trends and their consequences,
- novel parallel architectures.

Taking into account the extremely rapid pace of changes in the field SCPE is committed to fast turnaround of papers and a short publication time of accepted papers.

INSTRUCTIONS FOR CONTRIBUTORS

Proposals of Special Issues should be submitted to the editor-in-chief.

The language of the journal is English. SCPE publishes three categories of papers: overview papers, research papers and short communications. Electronic submissions are preferred. Overview papers and short communications should be submitted to the editor-in-chief. Research papers should be submitted to the editor whose research interests match the subject of the paper most closely. The list of editors' research interests can be found at the journal WWW site (`http://www.scpe.org`). Each paper appropriate to the journal will be refereed by a minimum of two referees.

There is no a priori limit on the length of overview papers. Research papers should be limited to approximately 20 pages, while short communications should not exceed 5 pages. A 50–100 word abstract should be included.

Upon acceptance the authors will be asked to transfer copyright of the article to the publisher. The authors will be required to prepare the text in LaTeX 2ε using the journal document class file (based on the SIAM's `siamltex.clo` document class, available at the journal WWW site). Figures must be prepared in encapsulated PostScript and appropriately incorporated into the text. The bibliography should be formatted using the SIAM convention. Detailed instructions for the Authors are available on the SCPE WWW site at `http://www.scpe.org`.

Contributions are accepted for review on the understanding that the same work has not been published and that it is not being considered for publication elsewhere. Technical reports can be submitted. Substantially revised versions of papers published in not easily accessible conference proceedings can also be submitted. The editor-in-chief should be notified at the time of submission and the author is responsible for obtaining the necessary copyright releases for all copyrighted material.