A publication of ISCA*:
International Society for Computers
and Their Applications

INTERNATIONAL JOURNAL OF COMPUTERS AND THEIR APPLICATIONS

TABLE OF CONTENTS

*** "International Journal of Computers and Their Applications is abstracted and indexed in INSPECT and Scopus."**

Volume 27, No. 2, June 2020 **ISSN 1076-5204**

International Journal of Computers and Their Applications

A publication of the International Society for Computers and Their Applications

ISCA Headquarters···•···278 Mankato Ave, #220, Winona, MN 55987···•···Phone: (507) 458-4517
E-mail: isca@ipass.net • URL: http://www.isca-hq.org

Guest Editorial:

Special Issue from ISCA Fall-2019 CAINE Conference

This special issue of IJCA is a collection of four refereed papers, three selected from The 33rd International Conference on Computer Applications in Industry and Engineering (CAINE 2020) and one from The 36th International Conference on Computers and Their Applications (CATA 2020). Each paper is an extension of one of the best papers submitted to the conferences. Both the conference paper and extended journal papers were reviewed by multiple reviewers, judging the originality, technical contribution, significance, and quality of presentation.

The papers in this special issue discussed a broad range of research topics in computer applications in industry and engineering. In the paper **"Evaluating the Microservice Architecture Style for Manufacturing Cell Controller Software"**, the authors surveyed several traditional architectures for information systems in manufacturing and pointed out Service-Oriented Architecture (SOA) and Reference Architecture Model Industry 4.0 are the future trend for the modern manufacturing. In addition, microservice architecture style was discussed to conquer the disadvantages of traditional SOA solution. Authors of the paper **"Mutual Fund Portfolio Management Using LSTM"** proposed a framework to apply Long Short-Term Memory networks on mutual fund portfolio. The proposed framework can be used to work with any number of business section and any number of shares for that particular sector. In an effort to make the traditional Distributed Hash Table(DHT)-based peer-to-peer (P2P) network more generic, the paper **"Generalization of RC-Based Low Diameter Hierarchical Structured P2P Network Architecture"** discussed how to use a modular arithmetic based mathematical model to design a two-level structured architecture. Lastly, in the paper **"A Detailed Comparison of the Effects of Code Refactoring Techniques in Different Mobile Applications"**, the authors provided a detailed evaluation of the impact of code refactoring techniques for energy efficiency and performance in mobile environments using GPS-UP metrics.

We would like to express our sincere appreciation to the contributions of all authors and reviewers to this special issue. We hope you will enjoy the special issue and look forward to seeing you at future ISCA conferences. More information about ISCA society can be found at http://www.isca-hq.org.

Guest Editors:

Gongzhu Hu, Central Michigan University
Yan Shi, University of Wisconsin-Platteville
Quan Yuan, University of Texas-Permian Basin

Evaluating the Microservice Architecture Style for Manufacturing Cell Controller Software

Christoph Wunck[*]
Emden/Leer University of Applied Sciences, Emden, GERMANY
Iowa State University, Ames, IOWA, USA
OFFIS Institute for Information Technology, Oldenburg, GERMANY

Jonas Kallisch[†]
Emden/Leer University of Applied Sciences, Emden, GERMANY

Abstract

This study elaborates on the advantages of migrating legacy IT systems for manufacturing operations to a microservice architecture, which is an important step towards a platform-based ecosystem. Traditional architecture models for manufacturing operations from the literature are evaluated. The different models' strengths are combined towards a common architecture for the factory of the future. Microservices are introduced as a new architectural style for manufacturing operations software.

1 Introduction

Manufacturers all over the world experience constant pressure in four areas: market-related pressure (e.g., customized products, market saturation), economy-related pressure (e.g., globalization of manufacturing, migration towards low cost economies), technology-related pressure (e.g., 3D printing, new materials, big data, security) and environment-related pressure (e.g., energy efficiency, sustainability, product life cycle) [1]. Many manufacturers are small or medium enterprises. For these smaller firms it is difficult to keep pace with the recent developments in *Industrie 4.0* and Smart Manufacturing, which are geared towards flexible manufacturing IT systems that can adapt easily to changing environments. They find it difficult to migrate to new technologies like cyber-physical production systems, machine-to-machine communication, or manufacturing execution and intelligence systems (MES) while sustaining their day-to-day manufacturing business.

Cyber-physical systems (CPS) have their origins in the area of embedded systems. A CPS may be defined as a computer system tightly bound to a mechanical system or physical process. An important application area of cyber-physical systems is smart manufacturing, where cyber-physical production systems (CPPS) cooperate across all levels of production. CPPS acquire information from their environment, act autonomously, connect to other systems or human operators and respond to internal and external changes. CPS and CPPS are expected to facilitate the creation of new business models and new services [16].

Figure 1 illustrates the functionality of manufacturing operations within a vertical stack of system levels. This layered architecture has been standardized by the international standard IEC 62264, also known as ISA-95. Manufacturing execution systems (MES) and other systems targeted to support manufacturing operations reside above the shop floor (level 0, 1, and 2) and beneath the enterprise resource planning level (ERP at level 4). The standard defines several categories of information models for the manufacturing operations management layer at level 3.

According to [14], MES will play a central role in manufacturing enterprises' path towards Industrie 4.0. On the other hand, MES has not been a widely explored concept in academic research. A migration path towards Industrie 4.0 for small and medium manufacturers must be both attractive regarding business opportunities and feasible regarding technological challenges. Investing in new technology is a business decision in the first place, and many of today's smart manufacturing technologies lack demonstrating a business case. The recent emergence of platforms and business ecosystems [9, 22], will have a major impact on manufacturing enterprises, as the MES seems to be a suitable candidate for the transition from a monolithic software system into a service- and platform-based ecosystem [25].

This paper elaborates on the advantages of migrating legacy IT systems for manufacturing operations to a microservice architecture, which is an important step towards a platform-based ecosystem. Section 2 evaluates traditional architecture models for manufacturing operations from the literature. Section 3 discusses how the different models may be combined towards a common architecture for the factory of the future. Section 4 introduces microservices as a new architectural style for manufacturing operations software. Section 5 describes a cyber-physical factory testbed utilizing Industrie 4.0 components, assets administrations shells and microservices.

[*] Department of Computer Science. E-mail: christoph.wunck@offis.de.
[†] E-mail: jonas.kallisch @hs-emden-leer.de.

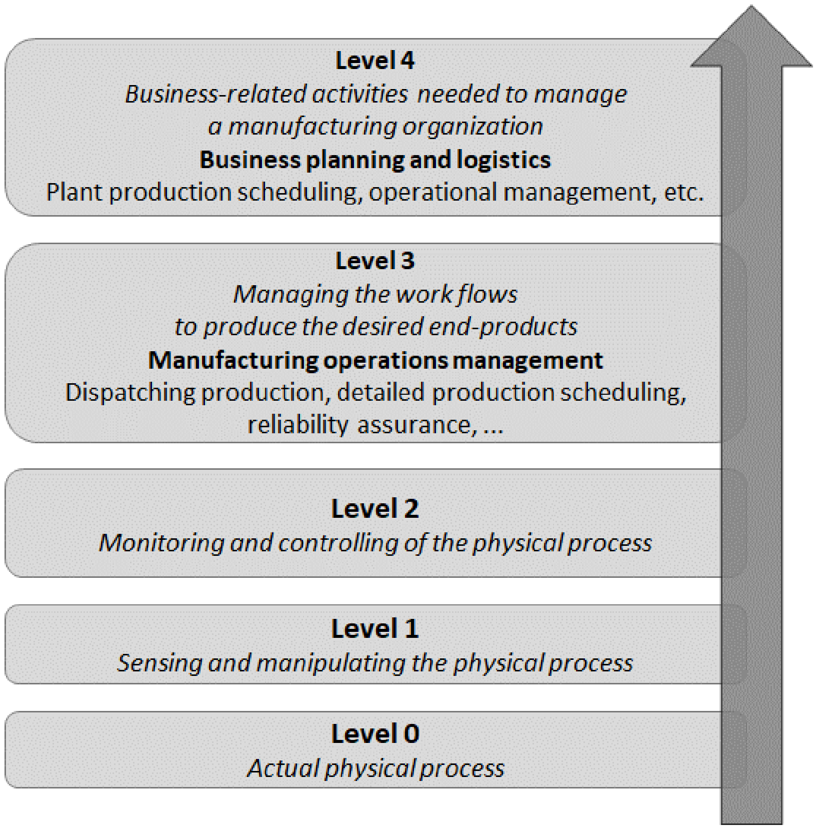

Level 4
Business-related activities needed to manage a manufacturing organization
Business planning and logistics
Plant production scheduling, operational management, etc.
Level 3
Managing the work flows to produce the desired end-products
Manufacturing operations management
Dispatching production, detailed production scheduling, reliability assurance, ...
Level 2
Monitoring and controlling of the physical process
Level 1
Sensing and manipulating the physical process
Level 0
Actual physical process

Figure 1: Vertical integration in the multi-level functional hierarchy of IEC 62264 [7]

Section 6 wraps up the conclusions.

2 Reference Architectures and Implementations

Manufacturing operations encompass a large set of activities like detailed scheduling, recipe management, resource management, production execution, work in progress management, production history or quality management, among others. Many manufacturers support their operations using a multitude of customized special-purpose applications and spreadsheet files [19].

Manufacturing Execution Systems (MES) are a type of application software designed specifically to support manufacturing operations. Typically these software systems are huge monoliths developed and marketed by a single vendor. It is difficult or costly for a manufacturer to have custom functions, algorithms or interfaces implemented if these are not contained in the standard set of MES functions.

Industrie 4.0 scenarios assume both smaller lot sizes to manufacture and less stable market conditions that manufacturers need to adapt to. A couple of software architecture patterns and implementations have been proposed by various authors or standardization bodies to accommodate the requirements of flexibility in manufacturing operations. This section discusses the most influential architectures known from the literature.

2.1 IEC 62264 / ISA-95

IEC 62264 is an international standard for enterprise-control system integration, based upon ANSI/ISA-95, which is a standard by the International Society of Automation (ISA). IEC 62264 reflects the hierarchical organization of enterprises by categorizing operations and activities in five levels. This level structure has its roots in the Purdue Enterprise Reference Architecture, developed in the early 1990s by Theodore J. Williams. Level 3 defines manufacturing operations in four areas, namely *production*, *maintenance*, *quality test*, and *inventory*. For each area of operations, four types of models, based on best practices, are defined [7]:

- Activity models describe what has to be done at the manufacturing level.
- Information models determine what types of entities are active at the manufacturing level, e.g., machines, workers, schedules. Models are elaborated as UML class diagrams.
- Data flow models show what types of data are exchanged during activities at the manufacturing level.
- Data structure models reveal how the internal structure of data looks like.

The definitions from IEC 62264 can be used to standardize the interfaces between ERP and MES systems, to improve communication in MES projects, and to develop interoperable software [19]. An XML implementation of the IEC 62264 models is given by the Business to Manufacturing Markup Language (B2MML). B2MML consists of a set of XML schemas that implement the data models in the standard. It is maintained by the Manufacturing Enterprise Solutions Association (MESA) XML Committee [15].

2.2 Holonic Manufacturing Systems

The concept of *Holon* was developed in the context of social organizations and living organisms to describe a whole-part-relationship between real-life objects. Holons are both self-contained entities to their subordinated parts, and dependent parts when seen from the inverse direction. H. Van Brussel, P. Valckenaer and others [23] adapted the concept of holons to manufacturing to attain benefits like stability during disturbances, adaptability and flexibility in changing environments, and efficient use of available resources. According to the authors, a Holonic Manufacturing System (HMS) preserves the stability of a hierarchical organization while providing the flexibility of a heterarchy. A holon in a manufacturing system is an autonomous and co-operative building block for transforming, storing, or validating information and physical objects. Therefore, a holon consists of an information processing part and often a physical processing part. A holon can be part of another holon. Holons can be modelled by UML objects and class diagrams. They can be part of a generalization hierarchy.

The authors of [23] proposed the HMS reference architecture PROSA (Product-Resource-Order-Staff Architecture). The acronym refers to the four types of holons:

- *Product holons* encapsulate knowledge about products and processes, e.g. bill of materials, process plans, or quality assurance procedures.
- *Resource holons* are abstractions of industrial assets, e.g. factory, machines, tools.
- *Order holons* represent tasks in the manufacturing system. They manage the physical product, its state and logistical information, e.g. customer orders, make-to-stock orders, orders to maintain and repair.
- *Staff holons* assist the aforementioned basic holons in performing their work. While basic holons are responsible for delivering their results and making their own decisions, the staff holons hand out advice. The concept of staff holons facilitates centralized functionality within the system, e.g. production schedules.

Figure 2 shows the three basic holons and their interaction. They exchange *process knowledge* on how to perform operations on resources, *production knowledge* on how to produce products, and *process execution knowledge* on how to execute process instances for certain customers.

Ongoing research on Holonic Manufacturing Systems targets areas of application and technical implementations. A Manufacturing Execution System based on PROSA was introduced in [21]. The research prototypes have been implemented as multi-agent systems.

The authors of [13] present an adaptive holonic control architecture (ADACOR) with four manufacturing holon classes: *product* holons, *task* holons, *operational* holons, and *supervisor* holons. The prototype was implemented as a multi-agent system based on the Java Agent Development Framework (JADE). An implementation of eye-tracking technology in Holonic Manufacturing Systems is described in [18].

2.3 Multi-Agent Systems

Multi-agent systems have been proposed as the preferred architecture for integrated manufacturing since the late 1990s [24]. A comprehensive summary of agent-based systems for

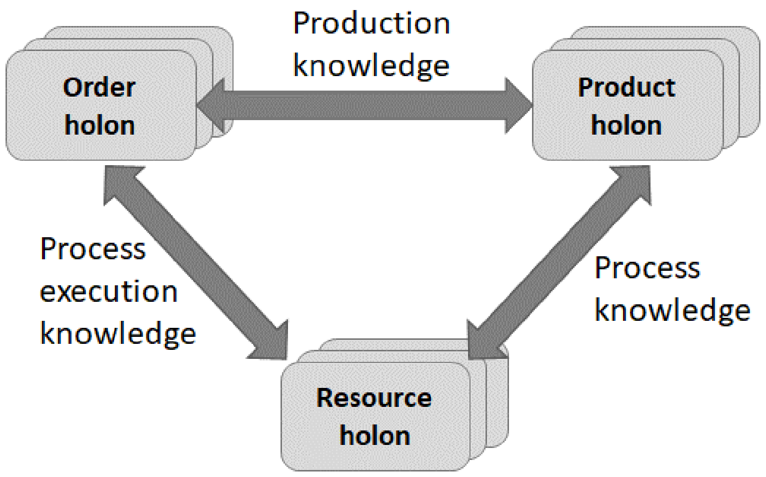


Figure 2: Basic building blocks of a Holonic Manufacturing System [23]

manufacturing is given in [17]. An agent is defined as a computational system that is situated in a dynamic environment and is capable of exhibiting autonomous and intelligent behavior.

An agent may have an environment that includes other agents. The community of interacting agents, as a whole, operates as a multi-agent system (MAS). Agent technologies target problems like agent behavior and decisions, interaction, organization, division of labor, coordination, and supervision. Holonic Manufacturing Systems are viewed as a special kind of MAS.

Many academic implementers use the Java Agent Development Framework (JADE) as their middleware platform. The software architecture is based on cooperating Java Virtual Machines. The communication between agents relies on Java Remote Method Invocation (RMI) and IIOP [2]. Agent services can be exposed as web services (WSDL). In-depth technological and architectural details about this framework are presented in [3].

A multi-agent implementation based on JADE for the holonic control of a manufacturing cell is presented in [10]. The authors conclude that MAS is a useful approach for the implementation of holonic architectures, as software agents and holons share several similarities. The authors of [12] focus on the comparison of multi-agent systems and IEC 61499 function blocks.

2.4 Service-oriented Architectures

Service-oriented architectures for industrial applications have been proposed by [10] in the context of the European Union FP7 project IMC-AESOP. The authors envision the factory of the future as a composition of services of varying complexity, and composed by other (potentially cross-layer) services. Services may be hosted either on field devices, local/edge servers or in the cloud.

The flat information-based architecture of the future coexists with the traditional hierarchical view from IEC 62264. Next generation industrial applications can rapidly be composed by selecting and combining new information and capabilities offered as services in the cloud. The collection of cloud services is targeted towards supporting IEC 62264 operations and activities.

The authors of [10] have elaborated several important requirements that need to be observed when implementing a service-oriented IT system for manufacturing operations support. In reference to architectural considerations, the following non-functional requirements are of particular interest:

- *Backward and forward compatibility*: Evolving infrastructure must not break existing functionality.
- *Combinability of services and tools*: Software applications will be created by a reuse-based software process according to [20].
- *Dynamic service discovery*: New devices or services announce their presence, while other components get aware of new capabilities.
- *Real-time interaction*: Technologies must meet the performance requirements of real-time interactions. Soft real-time essentially means low latency in the processing pipelines.
- *Technology-agnostic infrastructure evolution*: Future systems need to be updated to evolving technologies and simultaneously maintain their functionality.
- *Interoperability and open exchange formats*: Systems and services will have to communicate vertically and horizontally between systems at remote sites.
- *Mobility support*: Future systems need to be accessed via mobile devices, must support mobile assets in manufacturing and facilitate the migration of services.
- *Infrastructure Services*: Service developers should rely on a comprehensive middleware stack to not have to code software from scratch.
- *Scalability*: It should be easy to scale the system to new usage profiles, balance loads or deliver a defined quality of service.

Details regarding the implementation of the service-oriented architecture can be drawn from the IMC-AESOP project website [5], e.g., services are implemented as web services using technologies like HTTP, XML, SOAP, and WSDL.

2.5 Reference Architecture Model Industrie 4.0 (RAMI 4.0)

RAMI 4.0 was developed in Germany by the *Platform Industrie 4.0* consortium and standardized internationally in 2017 [6]. RAMI 4.0 references a couple of preexisting standards and enforces their application. The standard focuses on the cooperation and collaboration of technical assets. An asset is defined as any tangible or intangible item possessing a value for an organization. Assets can be as small as a screw and as extensive as an MES or ERP system. Assets have a life cycle and must be described in three different views or dimensions.

An *architectural view* describes the structure of an asset in terms of layers, similar to stacked layers of software systems. Interactions occur between adjacent layers. The top layer describes the asset's business context, the bottom layer corresponds to the real physical asset.

A *life cycle view* describes an asset from design, creation, usage and value generation, until phase-out and disposal.

A *hierarchy view* describes an asset's function in the context of a manufacturing site. The hierarchy is modeled after IEC 62264 and extended downwards to include the product itself as a value-adding part in the manufacturing value chain and upwards to include the connected world with cooperation between factories.

The standard illustrates the three views in the form of a three-dimensional layered cube, as depicted in Figure 3. The scope of manufacturing operations corresponding to IEC 62264 level 3 is indicated by red dotted lines.

The RAMI architecture model envisions manufacturing sites as a collection of so-called *Industrie 4.0 Components*. These are worldwide uniquely identifiable entities made of an asset and a software representation called *administration shell*. Industrie 4.0 components can be composed from other components and

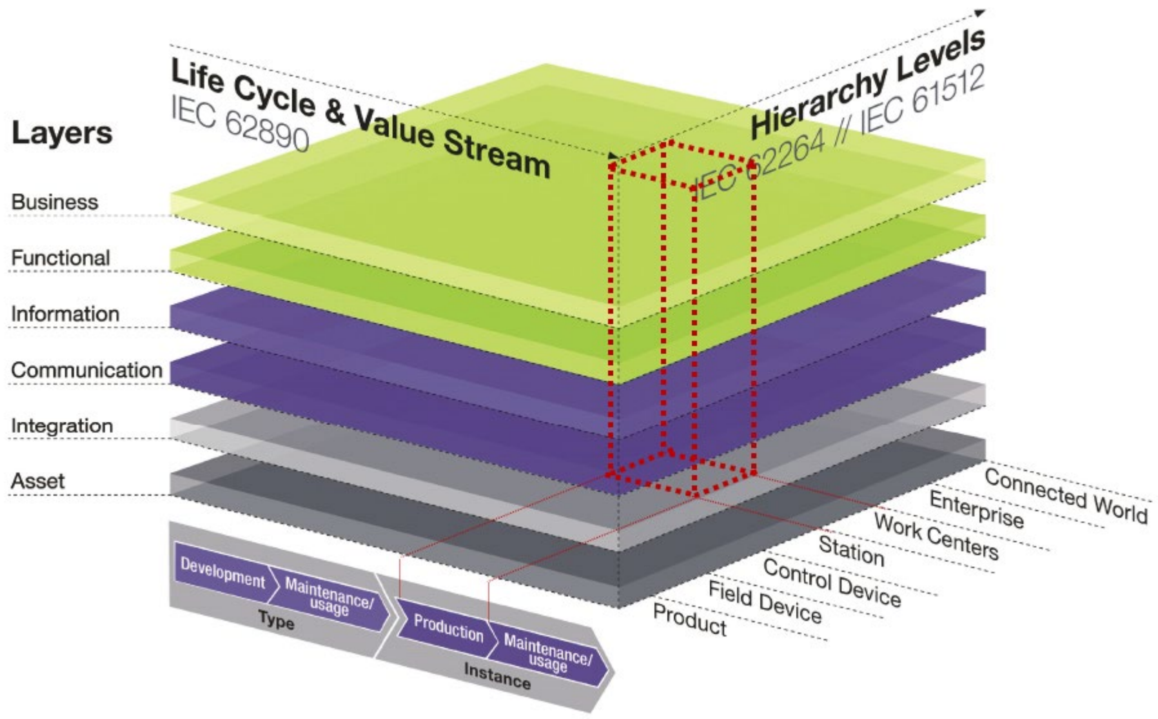


Figure 3: Scope of manufacturing (level 3) operations within RAMI [6]

can themselves be used to form more complex components. A component's attributes and state may be queried via software interfaces in the administration shell.

RAMI is designed to resolve basic interoperability issues in manufacturing and thus lays the foundation for the development of complex cyber-physical production systems (CPPS) that autonomously exchange information, trigger actions, and control each other independently [16]. The authors of [8] elaborate on the concept of an adaptive MES querying the administration shell of relevant assets in order to collect enough information about the manufacturing environment to adapt itself to changes.

3 Discussion of Traditional Architectures

The previous section presented short summaries of different architecture models for manufacturing operations from the literature. This chapter elaborates on how the respective strengths of these models can be combined to complement each other.

IEC 62264 partitions the application domain of manufacturing operations into smaller blocks, thus supporting the definition and validation of software requirements by domain specialists. The hierarchical model of IEC 62264 maps the division of labor prevalent in many successful enterprises. The level system has demonstrated its usefulness for more than 20 years and will be a part of coming solutions for the factory of the future, due to being one of the major dimensions of the RAMI 4.0 model. RAMI 4.0 is the foundation standard to ensure interoperability between assets. It will have a profound influence on the software landscape in industrial automation. However, many aspects important to smart manufacturing applications are not addressed by this very basic standard and still left to the decision of implementers.

The service-oriented architecture (SOA) proposed by [10] is an important step towards more architectural flexibility. Though the authors primarily focus on cloud-based services, the concept of SOA can easily be applied to local services on premises. The services offered might correspond to parts of an administration shell to access Industrie 4.0 components. The flexibility to design applications and to run them on a variety of software platforms will probably be a major incentive for manufacturers to abandon their customized legacy software and migrate to service-oriented architectures. The technology to implement SOAs varies in the course of time. Web services based on XML, SOAP and WSDL recently have lost some acceptance in the software engineering community due to low performance and high complexity in favor of microservices [4].

Service-oriented architectures are a technical requirement to create economic platform ecosystems. These will create completely new business opportunities for startups, small and medium enterprises both from manufacturing and ICT. A platform *owner* provides the infrastructure and rules for a marketplace that brings together *producers* and *consumers* of services [22]. *Providers* grant access to a platform via interfaces. In the manufacturing industry, machines, operators, or management might be both producers and consumers of services, depending on their respective role within the processes. Providers might be retrofitting devices, mobile phones, or networking interfaces like OPC UA. Platform owners might be today's vendors of MES, newcomers from related industries, or open source solutions without explicit ownership.

Holonic manufacturing systems (HMS) and multi-agent systems (MAS) have much in common. As mentioned before, many existing HMS implementations use an MAS approach. MAS oftentimes are implemented using the JADE middleware. Holonic manufacturing strives for highly autonomous modes of operation, which might collide with regulations in manufacturing regarding quality certification, safety procedures or legislation. HMS/MAS however might contribute to factories of the future in certain unregulated areas, where better alternatives to human decisions are welcome. In the context of a service-oriented architecture, HMS/MAS might even provide sophisticated services like planning and analytics, without interfering with highly regulated processes. Many technical implementations of MAS are bound to the JADE middleware, which is described by its maintainers as a niche technology far from mainstream software engineering [3]. This might be one of the obstacles that keeps small and medium manufacturing enterprises away. However, as the JADE runtime can be embedded into a hosting Java application, MAS applications based on JADE could be easily integrated into an SOA landscape.

4 Microservice Architecture

Microservices are the second generation of service-oriented architectures. They emphasize the development of highly maintainable and scalable software by decomposing large systems into sets of independent services with distinct business capabilities [4]. While the first generation of web services focused on complex technologies like syntactic service descriptions, discoverability and message contracts using XML Schema and Web Service Description Language (WSDL), the second generation aims to remove complexity in favor of simple functionalities. N. Dragoni, et al. [4] gives an in-depth overview of the evolution of distributed and service-oriented computing from an academic viewpoint.

The authors of [4] define a monolith as a software application whose modules cannot be executed independently, while a microservice is considered a cohesive, independent process interacting via messages. They confront six major issues of monoliths with corresponding microservice solutions:

1. *Evolution*: Large monoliths are difficult to adapt to new requirements. Microservices implement a limited functionality with a small codebase, keeping maintenance costs low.
2. *Dependencies*: Monoliths depend on numerous external libraries spread over different places by the operating systems. These dependencies may lead to versioning conflicts. Microservices can gradually evolve to a new version, while a previous version of the same service remains available as long as required.
3. *Deployment*: Deploying small changes in a monolith requires restarting the whole application. Microservices require only themselves to be restarted, resulting in short redeployment times.
4. *Runtime environment*: Deployment environments for monoliths cannot allow for all requirements of all modules equally. Microservices are run from containers providing an environment optimally tailored to the microservice's task.
5. *Scaling out*: Increasing load on a monolith is compensated by distributing the load among multiple instances of the whole monolith, which might not address the load problem efficiently. Multiple instances of microservices can address an increasing load in finer detail.
6. *Technology lock-in*: Evolution of a monolith is bound to the original language and technology. Microservices can migrate to new technologies any time during evolution, as long as the communication interfaces remain stable.

The services proposed by [5] as well as the service export mechanism of the JADE middleware belong to the first generation of web services requiring monolithic application servers for their execution. Due to their advantages over first-generation web services the authors of this study envision that microservices will be the technological foundation of service-oriented architectures for manufacturing operations. The requirements posed in Section 2.4 on SOA in manufacturing settings can be met by microservice implementations:

- *Backward and forward compatibility*: Microservices are designed to smoothly support both evolution of implementations and interfaces.
- *Combinability of services and tools*: This is a general property of SOA and therefore applies to microservice architectures alike.
- *Dynamic service discovery*: Service discovery as such is not part of microservice technology, but is supplied by middleware stacks like *Vert.x*, *Spring Boot*, and others.
- *Real-time interaction*: Microservices are designed for low latency.
- *Technology-agnostic infrastructure evolution*: Micro-services avoid technology lock-in and allow easy migration to technological innovations.
- *Interoperability and open exchange formats*: Message exchange is based on open protocols and formats like Hypertext Transfer Protocol (HTTP), XML and JSON.
- *Mobility support*: Current patterns for mobile application development expect microservices at the backend. Microservices are mainstream technology for this kind of software.
- *Infrastructure Services*: A choice of middleware stacks is available.
- *Scalability*: Microservices have been designed explicitly with respect to scalability.

Thus, the microservice architecture style combines the advantages of traditional service-oriented architectures with modern principles of software engineering geared towards rapid evolution and deployment cycles.

5 Case Study

The following case study from OFFIS Institute for Information Technology in Oldenburg, Germany, utilizes a microservice architecture to manufacture a variety of individualized products within a cyber-physical factory testbed. Figure 4 shows a schematic diagram and a photograph. The manufacturing cell comprises a 3D printer, laser cutter, collaborative robots, conveyor belts, and a manual assembly station.

The cell architecture conforms to the RAMI 4.0 framework. Each manufacturing device is represented as an Industrie 4.0 component and encapsulated by an asset administration shell (AAS). Each asset's functionality can be accessed by a corresponding microservice. Assets may be composed of subordinate assets to increase adaptability and maintainability. For example, the laser cutter device comprises the door, the extractor fan, and the cutter, each of which are represented by an AAS of its own. The subcomponents' microservices are orchestrated by a higher-level component.

As each microservice runs in a separate operating system process, services can be updated or replaced without stopping the complete system. The overall state of a microservice is described by a finite state machine. Figure 5 illustrates details of a sample interaction between a high-level orchestrating component and several subordinate components. The "Laser-Cutter-Service" on the left represents a higher level, orchestrating component, while the "Laser Cutter" in the middle depicts the cutter subcomponent of the laser cutter. The components' functions can be accessed via *skills*, which are implemented as microservices. Skills are comprised of methods (white boxes), which can be invoked via OPC-UA.

The manufacturing process is started by the higher level "Laser-Cutter-Service". This service has a skill called "Cutting-Skill", the current state of which is "Cutting". The service also has an "Engraving-Skill", which is not shown in Figure 5. After receiving the manufacturing order, the "Laser-Cutter-Service", i.e., the orchestrator, sends a message to the "Laser-Cutter" subcomponent to have it open the door. The subcomponent's state changes to "Open". Then the orchestrator instructs the "Industrial Robot" component to fill material into the working area of the laser cutter. Following this, the state of the "Laser-Cutter" subcomponent switches to "Filled" and the door is closed. Subsequently, the "Cut Skill" of the "Laser-Cutter" component is invoked. Finally, the orchestrator triggers the opening of the cutter device and initiates the evacuation of the finished part by invoking the appropriate skill of the "Industrial Robot" component.

6 Conclusions

This study examined several traditional architectures for information systems in manufacturing: Hierarchical Enterprise and Control Systems, Holonic Manufacturing Systems, Multi-Agent Systems, Service-Oriented Architecture (SOA), and Reference Architecture Model Industrie 4.0. The respective strengths of each architecture was acknowledged. It was shown that an SOA will likely be the foundation for the factory of the future, and aspects of each architecture under consideration could be accommodated within an SOA. The microservice architecture style was shown to remedy various deficiencies of earlier SOA middleware implementations. Future work will be directed to evaluate a use case based on the injection molding monitoring application presented in [24].

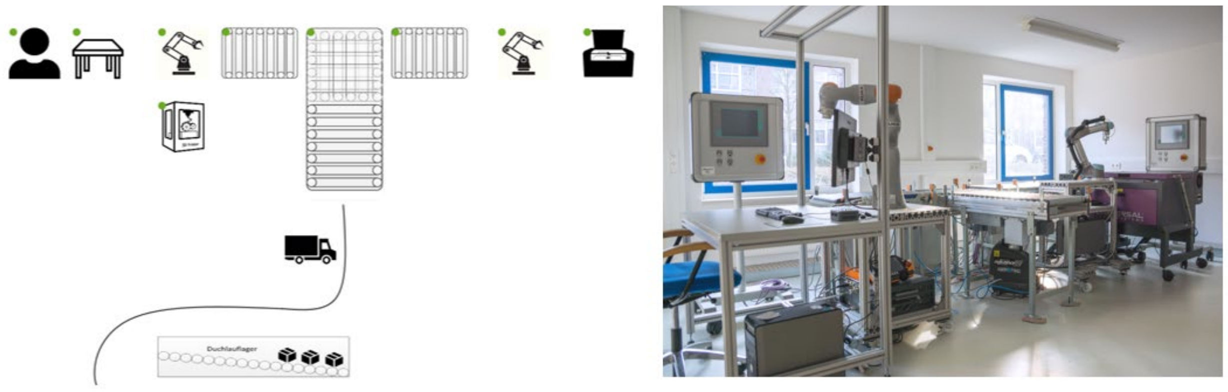


Figure 4: Cyber-physical factory testbed at OFFIS

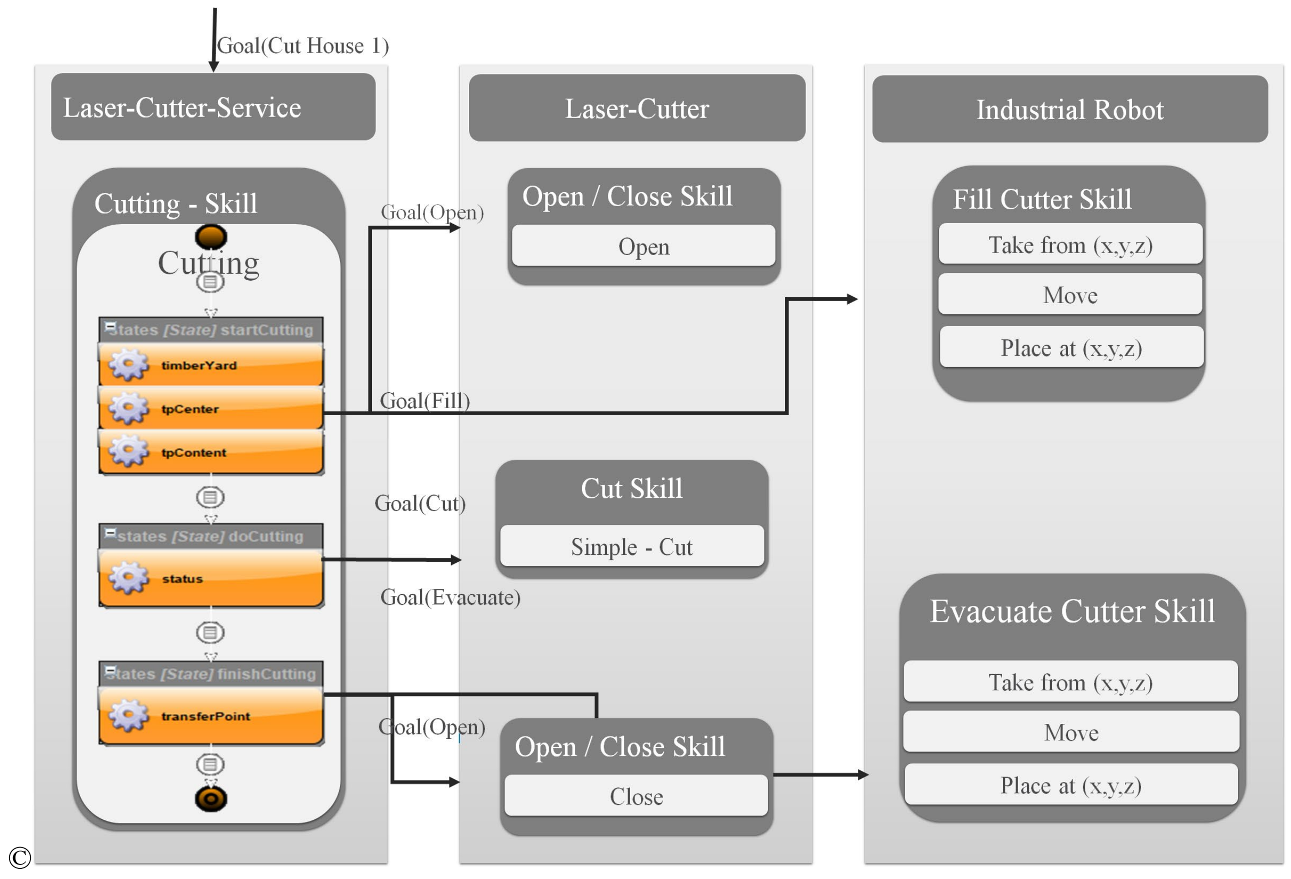


Figure 5: Example microservice architecture

References

[1] M. Abramovici, F. Bellalouna and J. C. Boebel, “Towards Adaptable Industrial Product-Service Systems,” *Proceedings of the 2nd CIRP IPS2 Conference 2010*, Linköping, Sweden, pp. 467-474, 2010.

[2] F. Bellifemine, G. Caire, G. Rimassa, A. Poggi, F. Bergenti, T. Trucco, D. Gotta, E. Cortese, F. Quarta, and G. Vitaglione, “General Questions about Jade,” [Online]. Available: https://jade.tilab.com/support/faq/. [Accessed: July 07, 2019].

[3] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa, “JADE: A Software Framework for Developing Multi-Agent Applications,” *Information and Software Technology*, 50:10-21, 2008.

[4] N. Dragoni, S. GiallorenziA. Lluch Lafuente M. Mazzara, F. Montesi, R. Mustain, and L. Safina, “Microservices: Yesterday, Today, and Tomorrow.” Present and Ulterior Software Engineering, pp 195-216, 2017.

[5] IC_AESOG Project, [Online]. Available: http://ww.imc-aesop.org/index.html. [Accessed: July 07, 2019].

[6] IEC PAS 63088, “Smart Manufacturing - Reference Architecture Model Industry 4.0 (RAMI 4.0),” 2017.

[7] IEC 62264, “Enterprise-Control System Integration,” 2015.

[8] J. Kallisch and F. Oppenheimer, “Adaptive Manufacturing Execution Systems (AMES): Best Practices for Implementations in Small and Medium-Sized Businesses,” *Proceedings of the 10th Annual European Decision Sciences Conference (EDSI)*, Nottingham, UK, pp. 1-12, 2019.

[9] R. Kapoor, “Ecosystems: Broadening the Locus of Value Creation,” *Journal of Organization Design*, 7(12):1-16, 2018.

[10] S. Karnouskos, A. W. Colombo, T. Bangemann, K. Manninen, R. Camp, M. Tilly, P. Stluka, F. Jammes, J. Delsing, and J. Eliasson, “A SOA-Based Architecture for Empowering Future Collaborative Cloud-Based Industrial Automation,” 38th Annual Conference on IEEE Industrial Electronics Society (IECON), Montreal, Canada, 2012.

[11] K. Kruger and A. Basson, “JADE Multi-Agent System Holonic Control Implementation,” Feb. 2018. Available: http://academic.sun.ac.za/mad/Papers/KrugerBasson_HolonicControl ImplementationUsingJadeMAS_20180208.pdf. [Accessed: July 07, 2019].

[12] K. Kruger and A. Basson, “Multi-Agent Systems vs IEC

61499 for Holonic Resource Control in Reconfigurable Systems," *Procedia CIRP*, pp. 503-508, July 2013.

[13] P. Leitão and F. Restivo, "ADACOR: A Holonic Architecture for Agile and Adaptive Manufacturing Control," Computers in Industry, pp. 121-130, Feb. 2006.

[14] S. Mantravadi and C. Møller, "An Overview of Next-Generation Manufacturing Execution Systems: How Important is MES for Industry 4.0?," *Procedia Manufacturing*, 30:588-595, 2019.

[15] MESA, "Business to Manufacturing Markup Language (B2MML)," [Online]. Available: http://www.mesa.org/en/B2MML.asp. [Accessed: July 07, 2019].

[16] L. Monostori, B. Kádár, T. Bauernhansl, S. Kondoh, S. Kumara, G. Reinhart, O. Sauer, G. Schuh, W. Sihn, and K. Ueda, "Cyber-Physical Systems in Manufacturing," *CIRP Annals - Manufacturing Technology*, 65:621-641, 2016.

[17] L. Monostori, J. Váncza and S. Kumara, "Agent-Based Systems for Manufacturing," CIRP Annals - Manufacturing Technology, Jan. 2006.

[18] J. Niemann, A. Basson, C. Fussenecker, K. Kruger, M. Schlosser, S. Turek, H. Umadevi Amarnath, "Implementation of Eye-Tracking Technology in Holonic Manufacturing Systems," *Procedia - Social and Behavioral Sciences*, 238:37-45, 2018.

[19] B. Scholten, *MES Guide for Executives*, Research Triangle Park, NC, USA: International Society of Automation, 2009.

[20] I. Sommerville, *Software Engineering*, Pearson Education, 2016.

[21] P. Valckenaers and H. Van Brussel, "Holonic Manufacturing Execution Systems," CIRP Annals, pp. 427-432, Jan. 2005.

[22] M. W. Van Alstyne, G. G. Parker, and S. P. Choudary, "Pipelines, Platforms, and the New Rules of Strategy," Harvard Business Review, April 2016.

[23] H. Van Brussel, J. Wyns, P. Valckenaers, L. Bongaerts and P. Peeters, "Reference Architecture for Holonic Manufacturing Systems: PROSA," *Computers in Industry*, 37:255-274, 1998.

[24] C. Wunck, "Implementation of Mobile Event Monitoring Agents for Manufacturing Execution and Intelligence Systems Using a Domain Specific Language," *Proceedings of the 2016 IEOM Conference*, Kuala Lumpur, Malaysia, 2016.

[25] C. Wunck and S. Baumann, "Manufacturing Execution Systems (MES) – The Next Platform Ecosystem," EDSI Annual Meeting, Udine, Italy, June 4, 2018.

Christoph Wunck is a full professor of Business Information Systems at Emden/Leer University of Applied Sciences. He is an Affiliate Professor in the Department of Computer Science at Iowa State University in Ames, Iowa, USA and serves as a Scientific Supervisor at OFFIS Institute for Information Technology in Oldenburg, Germany. He holds a doctoral degree from the Faculty of Mechanical Engineering and a master's degree (Diplom-Ingenieur) in Electrical Engineering, both from RWTH Aachen University

Jonas Kallisch is a Research Associate at Emden/Leer University of Applied Sciences. He holds a master's degree in Engineering & Management and a bachelor's degree in Information Systems, both from Jade University of Applied Sciences (Wilhelmshaven/ Germany).

Mutual Fund Portfolio Management Using LSTM

Achyut Ghosh, Soumik Bose and Soumya Sen [*]
University of Calcutta, Saltlake, Kolkata, INDIA

Giridhar Maji[†]
Asansol Polytechnic, Asansol 713302, West Bengal, INDIA

Narayan C. Debnath[‡]
Eastern International University, Binh Duong, VIETNAM

Abstract

Stock market prediction is one of the most difficult computations due to the many internal as well as any number and type of external factors. It is impossible to get the exact computation hence we look for the method which gives the computation with less error. Different machine learning methods are being applied for the computations which involve many parameters. In this research work we choose Long Short-Term Memory (LSTM) for the prediction as it is computationally suitable for these types of data analysis. After doing the prediction of share price the work is extended to manage portfolio of the mutual fund. The framework has been designed in such a way so that the portfolio manager can choose any number of business sectors as well as any number of shares belong to this sector. This research work henceforth applicable for computing individual share price as well as managing a diversified portfolio.

Key Words: Stock price prediction; LSTM; indian stock market; hybrid mutual fund; investment portfolio management.

1 Introduction

Data analytics is inevitable in any business applications for decision making, forecasting and prediction. Starting from financial sector, banking, share market, retail management, healthcare, HRMS (Human Resource Management System) etc. the analytics capability of data processing is explored for business decision making. Actually, any system which is complex and where many data parameters are used, data analytics are applied in the form of machine learning for decision making. Share market, which is a very important sector for financial application is an area where many parameters play crucial roles to predict the share price of a company. In this domain uncertainity and unpredictability are involved which makes the computation process challenging, therefore, analysis is required incorporating many of the parameters to take the correct decision. Machine Learning Techniques are suitable for these types of complex analyses involving many data parameters. Different techniques such as Artificial Neural Network (ANN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) etc. have been evolved in the area of Machine Learning for complex data analysis. Artificial Neural Network (ANN) is capable of learning from the historical data and helps in decision making. The methods such as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) etc. works great with multivariate time series data. One remarkable drawback of RNN is the vanishing gradient problem where with a large number of steps and backpropagation, the learning becomes negligible. LSTMs solves the vanishing gradient problem and also support arbitrary length time steps.

The people with more appetite for higher income often plan to invest in share market but refrain from investment due to the risk factors associated with share market. Due to that reason many such investors choose mutual funds as the alternative way to invest in the share market. Mutual funds are offered by banks, NBFC (Non Banking Financial Corporation), brokerage companies etc. It is being managed by experienced portfolio managers who have vast experience in the area of the stock market. These portfolio managers invest in multiple stocks with some specific goal of return (some with high return expectation but having high risk, some with lower return but less risk etc.). Investors instead of investing in specific stocks invest in these mutual funds. It is being found that right now these retail investors are investing more in mutual funds than investing in stocks directly. Therefore managing a mutual fund is the further extension over predicting the individual stock. In this research work we extend the previous work [9] of individual share prediction using LSTM to mutual fund portfolio management. There are many ways to form mutual funds. The mutual funds could be diversified where the investments are done in different sectors (such as banking, finance, retail, auto, auto-ancillary, manufacturing, FMCG, IT etc.). It could be sector specific (investors want to invest money in a specific sector only). Along with this another important factor to consider is time period. The investor has to specify the period for which he is investing the money. Portfolio managers will try to maximize the profit based on the investment period of the investor.This research work will consider both the types of the mutual funds and the

[*]A.K. Choudhury School of I.T., Email: achyutghosh06@gmail.com; 1994bolusoumik@gmail.com and iamsoumyasen@gmail.com

[†]Dept. of Electrical Engineering, Email: giridhar.maji@gmail.com

[‡]Department of Software Engineering, Email:narayan.debnath@eiu.edu.vn

time period.

2 Related Studies

Fama in 1970 [20] proposed efficient market hypothesis which says that in an efficient market (where all events are known to all stakeholders as when it happens) the effect all market events are already incorporated in stock prices, hence it is not possible to predict using past events or prices. But there is lots of research work in stock market prediction based on either fundamental analysis of the underlying economic factors or based on technical analysis that considers the historical prices. In fundamental analysis many different macro-economic factors are considered for a long term prediction [21]. In technical analysis, short term or medium term price predictions are made using different technical attributes of time series price history. Some of the attributes are day's opening price, closing price, average price, moving average etc. Many statistical data mining tools have been employed in such kind of prediction such as linear models like *AR, MA, ARIMA, ARMA, CARIMA*, etc. [6, 20] or non-linear models (*ARCH, GARCH, ANN, RNN, LSTM*, etc.). Recently, many soft computing heuristic techniques are also being used in financial prediction [27] with success. All of these are possible due to the availability of a large corpus of financial data in digital format which is clean and authentic and its sheer volume allows to create enough training dataset for the ANNs.

Researchers in [19] used a data mining technique known as *frequent itemset mining* to predict the movement of a sectorial index depending on a lagged correlation to other indices. They used a total of six sectoral indices from the Bombay Stock Exchange and considered a varying lag of 1 day to 5 day and analyzed the co-movement patterns using apriori algorithm as well as correlation coefficient.

Roondiwala et al. [23] employed an *RNN-LSTM* hybrid model on NIFTY listed scrips with four attributes of historical price such as every day's opening and closing price and the maximum and minimum price. They used a three week window to predict the following day's price movement.

The effectiveness of LSTM networks trained with backpropagation through time for stock price prediction is explored in [16]. Authors constructed many different LSTM architectures, trained and tested them. Authors in [29] used the *Naive Bayesian* emotional classifier on discussion forum data along with an LSTM time series learning model to improve the prediction accuracy while forecasting opening stock price on the Shanghai Composite index. A similar approach is employed in [1] where both numerical and textual data inputs are considered in predicting opening stock prices by converting media reports into a distributed representations using a *Paragraph Vector* and temporal effects of past events of selected companies with LSTM.

Kim et al. [17] used a *convolutional neural network*(CNN) to learn the features from the images of stock charts. The candlestick charts emerged as the best candidate for the CNN training image in predicting future stock movement. Then, they used an LSTM with stock price time series data. They tested with a minute-wise price and used a thirty minute overlapping window to predict the 35^{th} minute price on S&P 500 ETF data. They first used the CNN and LSTM separately on different representations of the same dataset and then used the combined feature fusion model. Experimental results indicate the superiority of the combined model in comparison to the individual models with a reduced prediction error. Authors in [7] used LSTM to predict directional movement of stocks on S&P 500 and compared the result with non-memory based models like Random forest, logistic regression and deep neural network. They observed that LSTM outperforms other predictive models on real stock directional movement data from 1992-2015 on S&P 500 listed stocks. Hansson in [10] compared direction of change classification on S&P stocks time series data using statistical models like AR, ARIMA to the LSTM prediction and concluded that LSTM outperforms in stock movement direction prediction while shown similar results with absolute price prediction.

Hiransha et al. [11], used three deep learning architectures such as RNN, CNN and LSTM to predict stock price using day wise past closing prices. They have considered two companies from the IT sector (TCS and Infosys) and one from the Pharma sector(Cipla) for experiment. The uniqueness of the study is that they trained the models using data from a single stock and then they used those models to forecast prices of five other stocks from NSE and NYSE (New York Stock Exchange). They argued that linear models try to fit the data to the model but in deep networks underlying dynamics of the stock prices are unearthed. As per their results CNN outperformed all other models as well as classical linear models. The DNN could forecast NYSE listed companies even though the model has learned from NSE dataset. The reason could be the similar inner dynamics of both the stock exchanges. The above results also corroborated by researchers in [25] with similar observations.

Gers & Schmidhuber proposed a variation of LSTM by introducing "peephole connections" [8]. In this model the gate layers can look into the cell state. In another case the model coupled forget and input gates. In this case, the decision to add new information or to forget it is taken together. It forgets only when it needs to input something in its place. This architecture inputs new values to the cell state when it forgets anything older. Cho et al. [5] proposed another popular LSTM variation known as the *Gated Recurrent Unit*(GRU). It aggregates both the forget and input gates into an "update gate". The cell state and hidden state are merged along with a few other minor modifications to make the final model simpler than the original LSTM. Due to the above reason this model is becoming popular day by day. These are by no means an exhaustive list of modified-LSTMs. There are many other variants such as *Depth Gated LSTM* by Yao et al. [28]. Authors in [2] used both the bidirectional LSTM and stacked LSTM predictive models in financial prediction for comparative evaluation. Koutnik et al. [18] proposed 'Clockwork RNNs' to tackle long-term dependencies in a completely different manner.

3 LSTM Architecture

In the following subsections we shall briefly introduce the recurrent neural networks with their advantages and limitations in predicting sequential data. Next, an improved version of RNN without the limitations of RNN in time series prediction, i.e. LSTMs are presented followed by detailed construction and working.

3.1 An overview of Recurrent Neural Network (RNN)

Classically, *neural networks* work with independent sets of input-output combinations of data. The output of one step is seldom fed into the input during the following step but a large family of real-world problems final output not only varies with external inputs, rather it depends on an earlier step output as well. For example, when humans read a book, understanding of each sentence depends not only on the current list of words but also on the understanding of the previous sentence or on the context that is created using past sentences. Humans don't start their thinking from scratch every second. As we go through this paragraph, our understanding of each subsequent word is based on the meaning of previous words. This concept of 'context' or 'persistence' is not available with classical neural networks. Inability to use context-based reasoning becomes a major limitation of traditional neural network. Recurrent neural networks (RNN) are conceptualized to alleviate this limitation of ANN while using sequential data. RNNs are networked with feedback loops within to allow persistence of information. Figure 1 shows a simple RNN with a feedback loop and its unrolled equivalent version side by side.

Initially, (at time step t) for some input X_t the RNN generates an output of h_t. In the next time step (t+1) the RNN takes two input X_{t+1} and h_t to generate the output h_{t+1} as shown in the below equation.

$$h_{t+1} = f(x_{t+1}, h_t)$$

A loop allows information to be passed from one step of the network to the next. RNNs are not free from limitations though. When the 'context' is from near past it works great towards the correct output. But when an RNN has to depend on a distant 'context' (i.e. something learned long past) to produce correct output, it fails miserably. This limitation of the RNNs was discussed in great detail by Hochreiter [12] and Bengio et al. [3]. They also traced back to the fundamental aspects to understand why RNNs may not work in long-term scenarios. The LSTMs are designed to overcome the above problem.

3.2 LSTM Networks

Hochreiter & Schmidhuber [15] introduced a special type of RNN which is capable of learning long term dependencies. Later on many other researchers improved upon this pioneering work in [4, 13, 22, 24]. LSTMs are perfected over the time to mitigate the long-term dependency issue. The evolution and development of LSTM from RNNs are explained in [14, 26]. Recurrent neural networks are in the form of a chain of repeating modules of the neural network. In standard RNNs, this repeating module has a simple structure like a single *tanh* activation layer as shown in Figure 2.

LSTMs follow this chain-like structure, however the repeating module has a different structure. Instead of having a single neural network layer, there are four layers, interacting in a very special way as shown in Figure 3.

In Figure 3, every line represents an entire feature vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denotes its content being copied and the copies going to different locations.

3.3 The Working of LSTM

In LSTM architecture, *LSTM cells* are used instead of commonly used hidden layers in other neural networks. The cells are composed of various gates that control the input data flow. An LSTM cell consists of the following, (i) input gate, (ii) cell state, (iii) forget gate, (iv) output gate along with (v) a sigmoid layer, (vi) tanh layer, and (vii) pointwise multiplication. The key to LSTMs is the *cell state*, the horizontal line running through the top of the diagram.The cell state is like a conveyor belt. This runs straight down the entire chain, having some minor linear interactions. The input gate is nothing but the data input vector x_t, and similarly the output gate consists of the output generated by the LSTM. The sigmoid layer outputs numbers between 0 and 1, describing how much of each component should be let through. A value of 0 means "let nothing through", while a value of 1 means "let everything through!" An LSTM has three of these gates, to protect and control the cell state. The first step of LSTM is to decide what information are to be thrown out from the cell state. It is made by a sigmoid layer called the *forget gate* layer. It looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1}. A 1 represents "completely keep this" while a 0 represents "completely remove this". Mathematically a forget gate is represented as shown in equation (1). The cell state is updated based on the output from other gates which are shown in equation 2, 3, 4 and 5.

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{1}$$

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{2}$$

$$C_t = \tanh(W_c.[h_{t-1}, x_t] + b_c) \tag{3}$$

$$O_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \tag{4}$$

$$h_t = o_t * \tanh(c_t) \tag{5}$$

In the next step it is decided what new information are going to be stored in the cell state. It has two parts. First, a sigmoid layer called the "input gate layer" decides which values are to

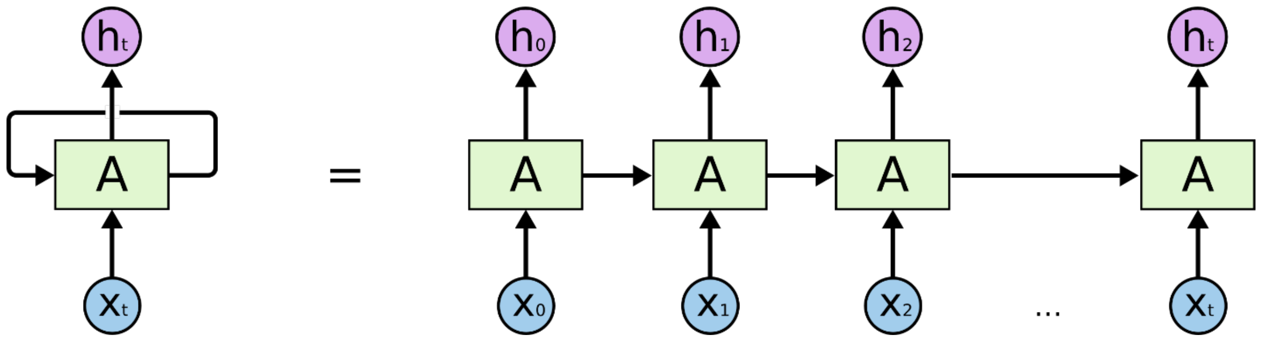


Figure 1: An unrolled recurrent neural network

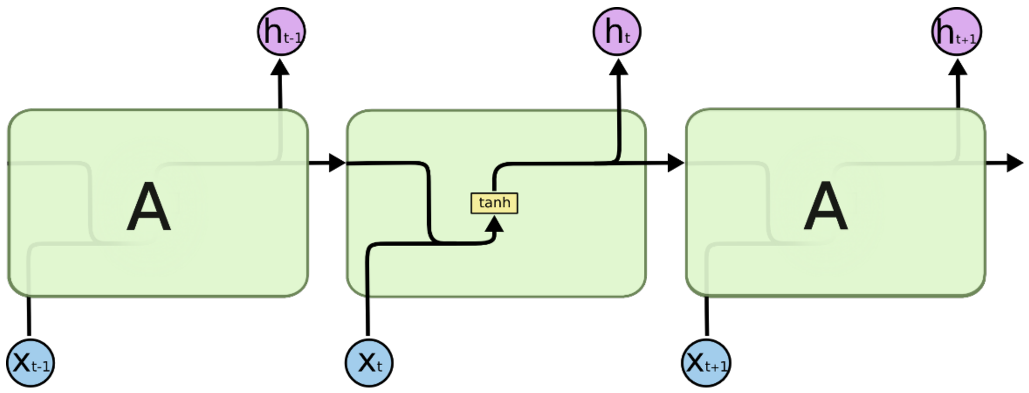


Figure 2: The repeating module in a standard RNN contains a single layer

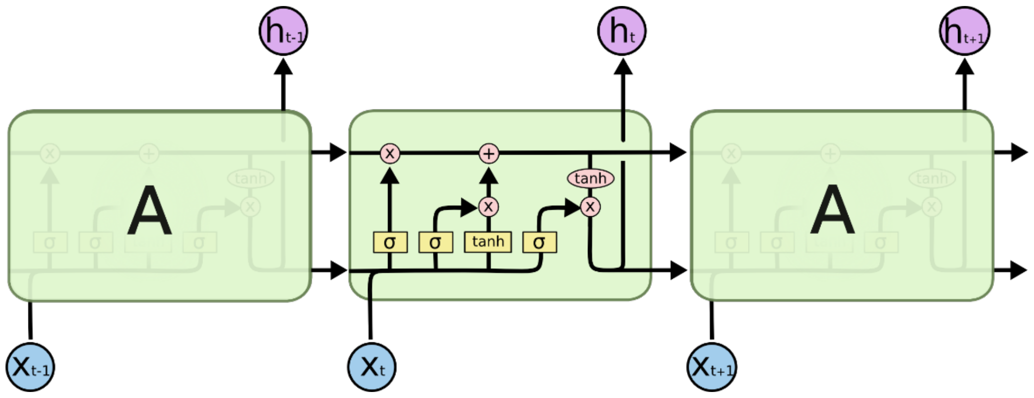


Figure 3: The repeating module in an LSTM contains four interacting layers

be updated. Thereafter, a tanh layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state. In the next step, these two are combined to create an update to the state. It is now time to update the old cell state, C_{t-1}, into the new cell state C_t. We multiply the old state by f_t. Then we add $i_t * \tilde{C}_t$. This is the new candidate value, scaled by how much we decide to update each state value. Finally, we need to decide on the output. The output will be a filtered version of the cell state. First, we run a sigmoid layer which decides what parts of the cell state we're going to output. Then, we put the cell state through *tanh* (to push the values to be between -1 and $+1$) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

4 Proposed Framework to Forecast Share Price & Company Growth in Different Time Spans

In this section at first we predict the share price for different time periods using LSTM by calculating the error value. Now based on the share values we proceed to form the mutual fund. If it is diversified then we choose the top performing shares of each segment for the given time period. Then we integrate them to form the mutual fund. If it is sector specific then we choose the top *k* shares (k is determined by portfolio manager) for the given time period and integrate them to form the mutual fund. In Section 4.2 we discuss about the share price prediction method as proposed in [9] and we extend this in Section 4.3 to form the mutual funds. Before that, in Section 4.1 we give an intuitive analysis why LSTM is chosen for analysis.

4.1 Analyzing Different Methods

Regression is one of the popular way to do the prediction in different business sectors including share market. In Figure 4 two figures on TCS share price using linear regression & polynomial regression of degree four are shown.

Regression is not always very useful to compute the error values. Similar problem also exist with curve fitting. The above graphs are showing a poor result in terms of curve fitting. It is found for time series data such as text, signals, stock prices, etc. LSTM is well suited to learn temporal patterns. LSTM is capable to solve the'vanishing gradient' problem that exists in a RNN while learning long-term dependencies with time series dataset with the use of memory cell (states) and (input and forget) gates. Therefore LSTM is chosen here for future prediction of the company's share price as well as growth.

4.2 Methodology to Compute Share Price

The methodology to compute the share price using LSTM was proposed in [9]. It is described here in brief. At first the future closing price of different companies are predicted from the historical price with the help of LSTM. This prediction is possible for any period. In our computation we have chosen the periods of 3-month, 6-month, 1 year, 2 year & 3 year. For these five different time spans (3-month, 6-month, 1 year, 2 year & 3 years) the growth of these companies are calculated. Then we analyze the deviations of closing price for each time span and from these we get the least error for the particular company. In our analysis, we have performed the analysis in the form of months. Then the weight of a company is defined as:

$$weight = \frac{2}{P*(P+1)}$$

In our case, month-wise weight (Y_i) will be calculated using the following algorithm:

Algorithm 1: Weight Calculation

```
$N := M$ $weight := \frac{2}{M*(M+1)}$
for $i = 1 : N$ do
    $Y_i := weight * N$ ;
    /* $Y_i$ is the weight of previous $i^{th}$ month*/
    $Q = Q - 1$;
    $i := i + 1$
```

Suppose the growth rate between different time periods is Gr_i where $i = 1 \ldots M$, considering current year as 0^{th} year. Therefore, Gr_i is the growth rate of $(i-1)^{th}$ time period w.r.t its immediate earlier year i.e. i^{th} year. To maximize the impact of current growth over the growth of an older year, we would develop a mathematical formula stated below. Suppose the growth rates of a company are Gr_1; $Gr_2 \ldots Gr_m$ respectively from present to M years earlier. Then the Company Net Growth Rate (CNGR) by the following formula.

$$CNGR_j = Y_1 * Gr_1 + Y_2 * Gr_2 + \cdots + Y_i * Gr_i + \cdots + Y_p * Gr_m$$

Where $CNGR_j$ is the Company Net Growth Rate of the j^{th} company (where $j = 1 \ldots m$)

4.3 Implementation Steps

Step 1: Raw Stock Price Dataset: Day-wise past stock prices of selected companies are collected from the BSE (Bombay Stock Exchange) official website.

Step 2: Pre-processing: This step incorporates the following:

a) Data discretization: Part of data reduction but with particular importance, especially for numerical data
b) Data transformation: Normalization.
c) Data cleaning: Fill in missing values.
d) Data integration: Integration of data files. After the dataset is transformed into a clean dataset, the dataset is divided into training and testing sets so as to evaluate.

Step 3: Feature Selection: In this step, data attributes are chosen that are going to be fed to the neural network. In this study, Date & Closing Price are chosen as the selected features.

Step 4: Train the NN model: The NN model is trained by feeding the training dataset. The model is initiated using random

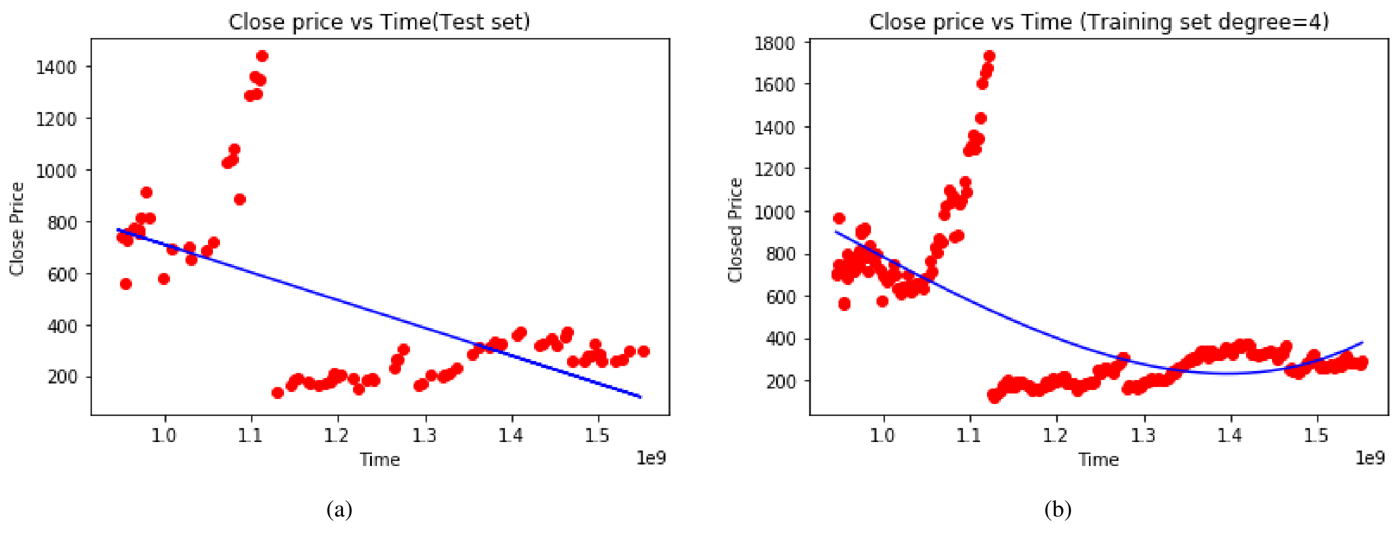


Figure 4: Stock market closing prices of TCS over a time period and polynomial(degree 4) regression line

weights and biases. The proposed LSTM model consists of a sequential input layer followed by 3 LSTM layers and then a dense layer with activation. The output layer again consists of a dense layer with a linear activation function.

Step 5: Output Generation: The RNN generated output is compared with the target values and error difference is calculated. The backpropagation algorithm is used to minimize the error difference by adjusting the biases and weights of the neural network.

Step 6: Test Dataset Update: Step 2 is repeated for the test data set.

Step 7: Error and Companies' Net Growth Calculation: By calculating deviation we check the percentage of error of our prediction with respect to actual price.

Step 8: Investigate Different Time Interval: We repeated this process to predict the price at different time intervals. Here we train 2-month dataset as training to predict 3-month, 6-month, 1 year, 2 years & 3 years of close price of the share. In this different time span, we calculate the percentage of error in the future prediction.

Step 9: We choose the month where we get the least error value and from that we compute the share price of each company.

(Mutual Fund Portfolio Formation Steps)

Step 10: (For Diversified Mutual Fund)

a) For different sectors choose the top performing company.
b) Invest the money equally in each top company of the selected sectors.

(For Sector Specific Mutual Fund)

a) Choose top k companies for that sector.
b) Select the investment ratio in each company as per the choice of portfolio manager or may equally be distributed in top k companies.

5 Results

The proposed system is implemented using Python. Here we consider 3 sectors namely IT, banking and pharmaceutical. In Table 1 the analysis is done for 3 year periods. Average closing price has been taken between 01.01.2012-31.12.2015. The price is estimated on 31.12.2019 that is after 4 years. Here from Table 1 we find *Monsanto* is the best performing company in IT sector.

Here from Table 2 we choose HDFC as the best performing company in the banking sector. Here in Table 3, we choose *CIPLA* as the best performing company in the pharmaceutical sector. It is to be noted as per our proposed methodology the error value prediction of the share price is not over 1.5%. It has been found that if we consider the test data for a longer period the error is less where as if the test data is for a short period then the error is high. Now we will show the formation of diversified mutual funds based on the proposed methodology. We have considered 3 sectors in the experiment and we found Monsanto as the top performing IT share with 60.42% growth, HDFC bank is the top performing banking share with 120.09% growth and CIPLA is the top performing pharmaceutical share with 17.89% growth. According to the proposed methodology equal amount of investments are done in every stock of the selected domain. Hence we get $(60.42 + 120.09 + 17.89)/3 = 66.13$ growth over the period of 4 years. The benchmark index of BSE, SENSEX has grown 57.95% in the same period (SENSEX was 26117 on 31.12.2015 and 41254 on 31.12.2019). Hence our proposed method gives better performance over SENSEX during the period. If we look at the annualized return we get 66.13/4=16.53% growth. In the Indian mutual fund sector this growth rate is among the well performing mutual funds. Generally the annual return over 12% is considered as good mutual funds. However continuous monitoring is required to exclude non-performing shares and include well performing

Table 1: Calculation of 4 Years for IT Sector

Company Name	Avg. Closing Price	Predicted Value	Error Percentage	% of Growth
TCS	1954.655	2353.04	0.134114252	20.54301143
Tech Mahindra	638.55	579.75	0.542107701	-8.713491504
Wipro	487.274	366.26	0.855394943	-24.18639205
Monsanto	1531.522	2453.74	0.128617363	60.42211604

Table 2: Calculation of 4 Years for Banking Sector

Company Name	Avg. Closing Price	Predicted Value	Error Percentage	% of Growth
SBI	158.75	272.118	1.313996134	71.41291339
HDFC	767.68	1689.635	0.211620853	120.0962641
ICICI	895.2	317.412	1.126491752	-64.54289544
AXIS	994.06	571.61	0.625534893	-42.49743476

Table 3: Calculation of 4 years for Pharmaceutical Sector

Company Name	Avg. Closing Price	Predicted Value	Error Percentage	% of Growth
CIPLA	471.404	555.78	0.162542913	17.8988723
Sun Pharmaceutical	744.874	581.97	0.155228105	-21.87000754
Lupin	1055.155	1091.32	0.082778745	3.427458525

shares.

In the same way calculation is done for other sectors also based on the top level companies belonging to that sector. The error values for the sector is shown in Table 2. It has been observed from the result that for almost all the sectors the error level comes down drastically with the test data for longer periods. So we suggest to apply this LSTM based model to predict the share price on long time historical data.

6 Conclusions

In this paper, the individual stock prediction is done for many periods to get the lowest error value so that the prediction is near to optimal. However as the investment in individual stock or individual sector is risky the previous analysis is extended for mutual fund portfolio management. The proposed framework is capable to work with any number of business sectors as well as any number of shares for that particular sector. Therefore the portfolio manager can tune the performance by adding or removing the number of sectors as well as shares from his portfolio. Results show that our proposed methodology gives better results than some of the existing well performing mutual funds in India. This research work can be extended further by using other different machine learning or deep learning techniques for better accuracy. Share market is very much unpredictable and therefore abrupt changes can take place at any moment due to many external factors (For example due to the *Coronavirus* outbreak in recent time the share market is going bearish all over the world). Hence continuous computation is required to find out what method is suitable at the current context for the best prediction.

References

[1] R. Akita, A. Yoshihara, T. Matsubara, and K. Uehara, ”Deep Learning for Stock Prediction Using Numerical and Textual Information”, In *2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICCIS), IEEE*, pp. 1–6, 2016.

[2] K. A. Althelaya, E.-S. M. El-Alfy, and S. Mohammed, ”Evaluation of Bidirectional LSTM for Short-and Long-Term Stock Market Prediction”, In *2018 9th International Conference on Information and Communication Systems (ICICS), IEEE*, pp. 151–156, 2018.

[3] Y. Bengio, P. Simard, and P. Frasconi, ”Learning Long-Term Dependencies with Gradient Descent is Difficult”, *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[4] J. Chen and N. S. Chaudhari, ”Segmented-Memory Recurrent Neural Networks”, *IEEE Transactions on Neural Networks*, 20(8):1267–1280, 2009.

[5] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation", *arXiv preprint arXiv:1406.1078*, 2014.

[6] Z. A. Farhath, B. Arputhamary, and L. Arockiam, "A Survey on ARIMA Forecasting Using Time Series Model", *Int. J. Comput. Sci. Mobile Comput*, 5:104–109, 2016.

[7] T. Fischer and C. Krauss, "Deep Learning with Long Short-Term Memory Networks for Financial Market Predictions", *European Journal of Operational Research*, 270(2):654–669, 2018.

[8] F. A. Gers and J. Schmidhuber, "Recurrent Nets That Time and Count", *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium, IEEE*, 3:189–194, 2000.

[9] A. Ghosh, S. Bose, G. Maji, N. C. Debnath, and S. Sen, "Stock Price Prediction Using LSTM on Indian Share Market", *Proceedings of 32nd International Conference on Computer Applications in Industry and Engineering (CAINE 2019), ISCA*, 63:101–110, 2019.

[10] M. Hansson, *On Stock Return Prediction with LSTM Networks*, PhD Thesis, Lund University, London, June 2017.

[11] M. Hiransha, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman, "NSE Stock Market Prediction Using Deep-Learning Models", *Procedia Computer Science*, 132:1351–1362, 2018.

[12] S. Hochreiter, *Investigations into Dynamic Neural Networks*, Master's Thesis, Diploma, Technical University of Munich, 1991.

[13] S. Hochreiter, "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions", *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[14] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory", *Neural Computation*, 9(8):1735–1780, 1997.

[15] S. Hochreiter and J. Schmidhuber, "LSTM Can Solve Hard Long Time Lag Problems", In *Advances in Neural Information Processing Systems*, pp. 473–479, 1997.

[16] H. Jia, "Investigation into the Effectiveness of Long Short Term Memory Networks for Stock Price Prediction", *arXiv preprint arXiv:1603.07893*, 2016.

[17] T. Kim and H. Y. Kim, "Forecasting Stock Prices with a Feature Fusion LSTM-CNN Model Using Different Representations of the Same Data", *PloS one*, 14(2):e0212320, 2019.

[18] J. Koutnik, K. Greff, F. Gomez, and J. Schmidhuber, "A Clockwork RNN", *arXiv preprint arXiv:1402.3511*, 2014.

[19] G. Maji, S. Sen, and A. Sarkar, "Share Market Sectoral Indices Movement Forecast with Lagged Correlation and Association Rule Mining", In *IFIP International Conference on Computer Information Systems and Industrial Management(CISIM)*, pp. 327–340, 2017.

[20] B. G. Malkiel and E. F. Fama, "Efficient Capital Markets: A Review of Theory and Empirical Work", *The Journal of Finance*, 25(2):383–417, 1970.

[21] D. Mondal, G. Maji, T. Goto, N. C. Debnath, and S. Sen, "A Data Warehouse Based Modelling Technique for Stock Market Analysis", *International Journal of Engineering & Technology*, 3(13):165–170, 2018.

[22] L. Pasa and A. Sperduti, "Pre-Training of Recurrent Neural Networks via Linear Autoencoders", In *Advances in Neural Information Processing Systems*, pp. 3572–3580, 2014.

[23] M. Roondiwala, H. Patel, and S. Varm, "Predicting Stock Prices using LSTM", *International Journal of Science and Research (IJSR)*, 6(4):1754–1756, 2017.

[24] J. Schmidhuber, D. Wierstra, M. Gagliolo, and F. Gomez, "Training Recurrent Networks by Evolino", *Neural Computation*, 19(3):757–779, 2007.

[25] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman, "Stock Price Prediction Using LSTM, RNN and CNN Sliding Window Model", In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE*, pp. 1643–1647, 2017.

[26] R. S. Sutton and A. G. Barto, *"Reinforcement Learning: An Introduction"*, MIT Press, 2018.

[27] S. Wichaidit and S. Kittitornkun, "Predicting Set 50 Stock Prices using CARIMA (Cross Correlation ARIMA)", In *2015 International Computer Science and Engineering Conference (ICSEC), IEEE*, pp. 1–4, 2015.

[28] K. Yao, T. Cohn, K. Vylomova, K. Duh, and C. Dyer, "Depth-Gated LSTM", *arXiv preprint arXiv:1508.03790*, 2015.

[29] Q. Zhuge, L. Xu, and G. Zhang, "LSTM Neural Network with Emotional Analysis for Prediction of Stock Price", *Engineering Letters*, 25(2):1–9, 2017.

Achyut Ghosh completed his Mater of Computer Application (MCA) from University of Calcutta in the year 2019 and obtained B.Sc (Honours) in Mathematics in the year 2016. His research interest is Machine Learning & Data Science. Currently he is practicing C#, Java based technology for development of industrial software.

Soumik Bose completed his Master of Computer Application (MCA) from University of Calcutta in the year 2019 and obtained B.Sc (Honours) in Mathematics in the year 2016. He is keen to solve research problems using Machine Learning technologies. His industrial expertization includes C#, Java script.

Giridhar Maji has been working as a Lecturer in the Department of Electrical Engineering, Asansol Polytechnic, Govt. of West Bengal, India since 2014. He had worked in the IT industry for more than 6 years in different capacities at Tata Consultancy Services and Cognizant Technology Solutions. He has done his M. Tech from University of Calcutta in 2015 and B. Tech from National Institute of Technology, Durgapur, India in 2007. He is currently working towards his PhD from University of Calcutta. He has published in more than 20 international journals and conferences. He has 4 book chapters to his credit. His area of research is data mining, data warehousing, social network analysis, information security and steganography. He serves as reviewer for Springer IEI Series-B journal since 2018 and also an editorial review board member of IGI Global International Journal of Information Security and Privacy (IJISP).

Narayan C. Debnath earned a Doctor of Science (D.Sc.) degree in Computer Science and also a Doctor of Philosophy (Ph.D.) degree in Physics. He is currently the Founding Dean of the School of Computing and Information Technology at Eastern International University, Vietnam. He is also serving as the Head of the Department of Software Engineering at Eastern International University, Vietnam. He has been the Director of the *International Society for Computers and Their Applications* (ISCA) since 2014. Formerly, He served as a Full Professor and Chairman of Computer Science at Winona State University, Minnesota, USA. He has been an active member of the ACM, IEEE Computer Society, Arab Computer Society, and a senior member of the ISCA.

Soumya Sen is an Assistant professor in A. K. Choudhury School of Information Technology under University of Calcutta. He has been awarded with Ph.D. in 2016 and obtained his M. Tech. Degree in Computer Science & Engineering in 2007 and M.Sc. in Computer & Information Science in 2005. He has around 80 research papers published in Peer reviewed journals and International Conferences. He has 3 patents registered in USA, Japan & South Korea. He has also published 2 books with Springer in the form of Springer Briefs. He is a reviewer of many International Journals and PC member & reviewer of many International conferences across the world. He is member of IEEE, ACM.

Generalization of RC-Based Low Diameter Hierarchical Structured P2P Network Architecture

Swathi Kaluvakuri*, Koushik Maddali*, Nick Rahimi†, Bidyut Gupta*
Southern Illinois University Carbondale, Carbondale, IL USA

Narayan Debnath‡
Eastern International University, VIETNAM

Abstract

In this work, we have considered generalizing a non-DHT-based low diameter hierarchical structured P2P network which is designed applying modular arithmetic, called the Residue Class (RC). This is an interest based two level architecture. At its second level, diameter of each cluster (group) of peers is one. It helps in designing very efficient data lookup algorithms. Besides, complexity involved in data lookup is a function of the number of distinct resource types unlike in DHT-based structured P2P systems. In addition, use of the same code to denote a resource type and the corresponding group-head has made the search process simple and efficient.

Key Words: Structured P2P network, residue class, network diameter, data lookup, generalization, interest based, churn.

1 Introduction

Peer-to-Peer (P2P) overlay networks are widely used in distributed systems due to their ability to provide computational and data resource sharing capability in a scalable, self-organizing, distributed manner. P2P networks are classified into two classes: unstructured and structured ones. In unstructured systems [2] peers are organized into arbitrary topology. It takes help of flooding for data look up. Problems arising due to frequent peers joining and leaving the system, also known as churn, is handled effectively in unstructured systems. However, it compromises with the efficiency of data query and the much needed flexibility. In unstructured networks, lookups are not guaranteed. On the other hand, structured overlay networks provide deterministic bounds on data discovery. They provide scalable network overlays based on a distributed data structure which actually supports the deterministic behavior for data lookup. Recent trend in designing structured overlay architectures is the use of distributed hash tables (DHTs) [13, 15, 20]. Such overlay architectures can offer efficient, flexible, and robust service [8, 13, 15-16, 20].

*School of Computing. E-mail: swathi.kaluvakuri@siu.edu, koushik@siu.edu, bidyut@cs.siu.edu.
†School of Information Systems & Applied Technologies. E-mail: shrahim@siu.edu.
‡School of Computing and Information Technology. E-mail: narayan.debnath@eiu.edu.vn.

However, maintaining DHTs is a complex task and needs substantial amount of effort to handle the problem of churn. So, the major challenge facing such architectures is how to reduce this amount of effort while still providing an efficient data query service. In this direction, there exist several important works, which have considered designing hybrid systems [1, 4, 7, 11, 14, 17-18]; these works attempt to include the advantages of both structured and unstructured architectures. However, these works have their own pros and cons.

1.1 Our Contribution

In this paper, we have considered interest-based P2P systems [3, 17, 19]. The rationale behind this choice is that users sharing common interests are likely to share similar contents, and therefore searches for a particular type of content are more efficient if peers likely to store that content type are neighbors [10]. We have pointed out above the disadvantages of DHT-based systems. Therefore, in this paper, we have considered the problem of designing and making it more generic by generalizing the non-DHT based structured system that does not have such problems [5, 12]. We have used a mathematical model based on modular arithmetic, specifically residue class (RC) [6, 9], to design a two-level structured architecture. To the best of our knowledge, there does not exist any such work that has used this mathematical model. Use of such a mathematical model has helped in designing very efficient data lookup algorithms. The proposed architecture has appeared in Preliminary Section 2. Besides, we have shown that the time complexity in searching for a resource is independent of the number of peers in the network and is instead bounded by the number of distinct resource types. In Section III, we have presented the Generalization and discussed about the Data lookup in Section IV and performance comparison in Section VI.

2 Preliminaries

Some of the preliminaries of this RC-based low diameter two level hierarchical structured P2P network [6, 9] have been considered here. In this section, we present a structured architecture for an interest-based peer-to-peer system. The following notations along with their interpretations will be used while we define the architecture.

Definition 1. *We define a resource as a tuple <R_i, V>, where R_i denotes the type of a resource and V is the value of the resource.*

Note that a resource can have many values. For example, let R_i denote the resource type 'movies and V' denote a particular actor. Thus <R_i, V'> represents movies (some or all) acted by a particular actor V'.

Definition 2. *Let S be the set of all peers in a peer-to-peer system. Then S = {P^{Ri}}, $0 \leq i \leq n-1$, where P^{Ri} denotes the subset consisting of all peers with the same resource type R_i. and the number of distinct resource types present in the system is n. Also, for each subset P^{Ri}, we assume that P_i is the first peer among the peers in P^{Ri} to join the system. We call P_i as the group-head of group G_i formed by the peers in the subset P^{Ri}.*

We now describe our proposed architecture suitable for interest-based peer-to-peer system. We assume that no peer can have more than one resource type. Generalization of the architecture comes in the next section.

2.1 Two Level P2P Hierarchy.

It is a two-level overlay architecture and at each level structured networks of peers exist. It is explained in detail below.

1) At level-1, we have a ring network consisting of the peers P_i ($0 \leq i \leq n-1$). The number of peers on the ring is n which is also the number of distinct resource types. This ring network is used for efficient data lookup and so we name it as transit ring network.
2) At level-2, there are n numbers of completely connected networks (groups) of peers. Each sub group, say G_i is formed by the peers of the subset P^{Ri}, ($0 \leq i \leq n-1$), such that all peers (ϵ P^{Ri}) are directly connected (logically) to each other, resulting in the network diameter of 1. Each G_i is connected to the transit ring network via its group-head P_i.
3) Each peer on the transit ring network maintains a global resource table (GRT) that consists of n number of tuples. GRT contains one tuple per group and each tuple is of the form <Resource Type, Resource Code, Group Head Logical Address>, where Group Head Logical Address refers to the logical address assigned to a node by our proposed overlay P2P architecture. Also, Resource Code is the same as the group-head logical address.
4) Any communication between a peer p_i' ϵ G_i and p_j' ϵ G_j takes place only via the respective group-heads P_i and P_j.

The proposed architecture is shown in Figure 1.

2.2 Relevant Properties of Modular Arithmetic

Consider the set S_n of nonnegative integers less than n, given as $S_n = \{0, 1, 2, \ldots (n-1)\}$. This is referred to as the set of residues, or residue classes (mod n). That is, each integer in S_n represents a residue class (RC). These residue classes can be labelled as [0], [1], [2], …, [n – 1], where [r] = {a: a is an integer, a ≡ r (mod n)}.

For example, for n = 3, the classes are:

$$[0] = \{\ldots, -6, -3, 0, 3, 6, \ldots\}$$
$$[1] = \{\ldots, -5, -2, 1, 4, 7, \ldots\}$$
$$[2] = \{\ldots, -4, -1, 2, 5, 8, \ldots\}$$

Thus, any class r (mod n) of S_n can be written as follows:

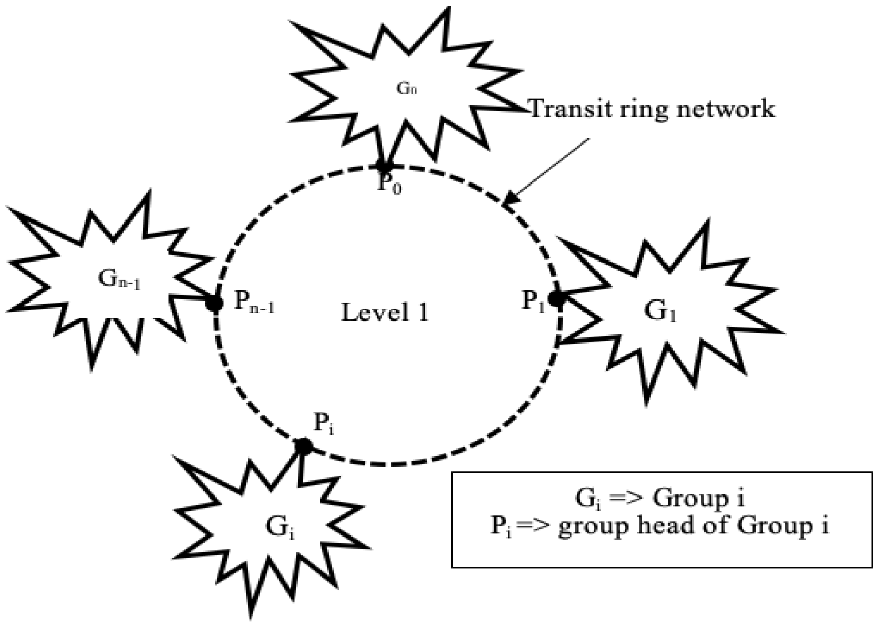


Figure 1: A two-level structured architecture with distinct resource types

[*r*] = {…., (r - 2n), (r - n), r, (r + n), (r +2 n), …, (r + (j-1). n), (r + j.n), (r + (j+1).n), …..}

A few relevant properties of residue class are stated below.

Lemma 1. *Any two numbers of any class r of S_n are mutually congruent.*

Proof. Let us consider any two numbers N' and N'' of class r. These numbers can be written as

$N' \equiv r \pmod n$; therefore, $(N' - r) / n$ = an integer, say I' (1)

$N'' \equiv r \pmod n$; therefore, $(N'' - r) / n$ = an integer, say I'' (2)

Using (1) and (2) we get the following,

$(N' - N'') / n = ((N' - r) - (N'' - r)) / n = I' - I''$ = an integer.

Therefore, N' is congruent to N''; that is, $N' \equiv N'' \pmod n$;

also, $N'' \equiv N' \pmod n$ because congruence relation ($\equiv$) is symmetric. Hence, the proof. □

2.3 Assignments of Overlay Addresses.

Assume that in an interest-based P2P system there are n distinct resource types. Note that n can be set to an extremely large value a priori to accommodate large number of distinct resource types. Consider the set of all peers in the system given as $S = \{P^{Ri}\}$, $0 \leq i \leq n-1$. Also, as mentioned earlier, for each subset P^{Ri} (i.e., group G_i) peer P_i is the first peer with resource type R_i to join the system.

In the proposed overlay architecture, the positive numbers belonging to different classes are used to define the following parameters:

1) Logical addresses of peers in a subnet P^{Ri} (i.e., group G_i). Use of these addresses will be shown to justify that all peers in G_i are directly connected to each other (logically) forming an overlay network of diameter 1. In graph theoretic term, each G_i is a complete graph.
2) Identifying peers that are neighbors to each other on the transit ring network.
3) Identifying each distinct resource type with unique code.

The assignment of logical addresses to the peers at the two levels and the resources happen as follows:

1) At level-1, each group-head P_r of group G_r is assigned with the minimum nonnegative number (*r)* of *residue class r (mod n)* of the residue system S_n.
2) At level-2, all peers having the same resource type R_r will form the group G_r (i.e., the subset P^{Rr}) with the group-head P_r connected to the transit ring network. Each new peer joining group G_r is given the group membership address (r + j.n), for j = 1, 2, …
3) Resource type R_r possessed by peers in G_r is assigned the code *r* which is also the logical address of the group-head P_r of group G_r.
4) Each time a new group-head joins, a corresponding tuple <Resource Type, Resource Code, Group Head Logical Address> is entered in the global resource table (GRT).

Remark 1. *GRT remains sorted with respect to the logical addresses of the group-heads.*

Definition 3. *Two peers P_i and P_j on the ring network are logically linked together if (i + 1) mod n = j.*

Remark 2. *The last group-head P_{n-1} and the first group-head P_0 are neighbors based on Definition 3. It justifies that the transit network is a ring.*

Definition 4. *Two peers of a group G_r are logically linked together if their assigned logical addresses are mutually congruent.*

Lemma 2. *Diameter of the transit ring network is n/2.*

Lemma 3. *Each group G_r forms a complete graph.*

Proof. According to Definition 4, two peers of a group G_r are logically linked together if their assigned logical addresses are mutually congruent. Also, from Lemma 1, we note that any two numbers of any class r of S_n are mutually congruent. Therefore, every peer has direct logical connection with every other peer in the same group G_r. Hence, the proof. □

2.4 Salient Features of the Overlay Architecture.

We summarize the salient features of this architecture.

1) It is a hierarchical overlay network architecture consisting of two levels; at each level the network is a structured one.
2) Use of modular arithmetic allows a group-head address to be identical to the resource type owned by the group. We will show in the following section the benefit of this idea from the viewpoint of achieving reasonably very low search latency.
3) Number of peers on the ring is equal to the number of distinct resource types, unlike in existing distributed hash table-based works some of which use a ring network at the heart of their proposed architecture. [15, 17].
4) The transit ring network has the diameter of n/2. Note that in general in any P2P network, the total number of peers N >> n.
5) Each overlay network at level 2 is completely connected. That is, in graph theoretic term it is a complete graph consisting of the peers in the group. So, its diameter is just 1. Because of this smallest possible diameter (in terms of number of overlay hops) the architecture offers minimum search latency inside a group.

2.5 Problem Formulation

In the architecture proposed above, it is assumed that no peer can have more than one resource type and this could be a very hard restriction practically. To overcome this restriction, we came up with the concept of *Generalization* i.e., the architecture is generalized in such a way that a peer can have multiple resource types. To implement this idea, we have redesigned the concept of peers joining a network in Section 3, the data lookup algorithms in Section 4. In Section 5, we have considered concurrent joins and leaves, and finally the maintenance of the transit ring network in Section 6.

3 Generalization of the Architecture

Generalization of the Architecture is dealt in multiple scenarios. Now, let us consider a situation that in group G_i, the group-head P_i or a peer p that belongs to G_i wants data insertion in the system of another existing resource type R_k.

3.1 Peer with Multiple Existing Resource Types

Scenario: Let us consider that in G_i the group-head P_i or a peer p (ϵ G_i) wants data insertion in the system of another existing resource type R_k,

Note that R_k exists in G_k and P_i or p already possesses R_i.

Solution: The solution for this scenario is as follows. The group-head P_i or peer p will now become a member of group G_k as well. So, it is understood that the IP address of P_i /p will be known to members of both the groups G_i and G_k. It logically means that, in the overlay network, Pi /p will be directly connected to all the members of G_i and G_k as well. The implementation of this proposal is stated below in Figure 2.

Data- Insertion Algorithm states the implementation

Time complexity of Algorithm 1 is bounded by (1+ n/2), n being the number of distinct resource types. Data insertion for more existing resource types can be done similarly.

3.2 Existing Peers Declaring New Resource Types

Scenario: Consider a P2P interest-based system which has S distinct resource types, viz., R_0, R_1, R_2, … R_{s-1}. Without any loss of generality, let us assume a scenario where peer P_i /p in G_i wants a data insertion for a new resource type R_s.

Note that R_s doesn't exist in system so far and P_i or p already possesses R_i. The implementation of this scenario is stated below.

Solution: As it's said before, each peer on the transit ring network maintains a global resource table (GRT) that consists of S number of tuples, where S is the number of distinct resource types in this case. GRT contains one tuple per group and each tuple is of the form <Resource Type, Resource Code, Group Head Logical Address>, where Group Head Logical Address refers to the logical address assigned to a node by our proposed overlay P2P architecture. So, based on this P_i can know that the new Resource type doesn't exist in the system so far.

Step 1: P_i /p will become the group-head of the newly created group G_s possessing resource type R_s

Step 2: The Transit-ring has to be reconstructed in the following way.

- As the recent group-head P_s, the new location of P_i or p on the ring is now between P_{s-1} and P_0
- If the peer declaring the new resource type is P_i, it will appear (logically) twice as group-heads on the ring for G_i and G_s
- If it is peer p, it will appear once as the group-head of G_s and once as a member of G_i.

Step 3: Note that the Resource code of the new Group head R_s is its logical address S.

Step 4: Now, P_i /p will ask the group-heads to update their GRT by including R_s and its code along with the IP address of P_i /p

Step 5: For implementation, P_i /p will now have another set of pointers pointing to its new neighbors, P_{s-1} and P_0.

Step 6: Group-heads P_{s-1} now changes its right neighbor from P_0 to P_s and group-head P_0 changes its left neighbor from P_{s-1} to P_s; they adjust their pointers accordingly.

<u>**Data-Insertion Algorithm**</u>

```
1.  Data insertion request for $R_k$ from $P_i$/p is forwarded along the transit ring from group-head $P_i$ to $P_k$
    // no. of hops along the ring in worst case is n / 2

2.  $P_k$ assigns to $P_i$/p the next available address, not yet assigned in group $G_k$
    // The new peer joining group $G_k$ is given the group membership address as (k + j.n)

3.  a. $P_k$ broadcasts the address of $P_i$/p in $G_k$
       // $P_i$/p is the new member of $G_k$ ; 1-hop communication
    b. Each group member of $G_k$ updates its list of neighbors

4.  $P_k$ unicasts a copy of neighbor list to $P_i$/p
    // $P_i$/p is a new member of $G_k$ now
```

Figure 2: Data insertion algorithm for multiple existing resource types

Next, we consider the data look up in the generalized structure.

4 Data Look-Up

Data lookup can be either intra-group or inter-group. The former one means that a peer p_i' (ϵ G_i) requests for some resource $<R_i,V''>$ which it does not possess. Note that only some peer(s) p_i'' (ϵ G_i) can possess $<R_i,V''>$ if at all; no other peer in any other group G_k can possess it since it is an interest based architecture.

The following data structure will be used for efficient data lookup. As mentioned earlier every group-head P_i will maintain a global resource table (GRT) with identical contents. We have earlier mentioned that the code of a resource type R_i is the same as the logical address of the corresponding group-head.

Apart from maintaining a GRT, each P_i maintains the following: each P_i has pointers to its two neighbors on the transit ring network. That is, each P_i knows the IP address of each of its two-neighboring group-heads P_{i-1} and P_{i+1}. The pointer information of P_i is also saved in the peer (ϵ G_i) with the next logical address. This saved information can be used to achieve fault tolerance in the event that P_i crashes or leaves. Each member peer in a group maintains a list of all its neighbors present in the group

4.1. Intra-Group Data Lookup

Without any loss of generality, let us consider a data lookup in group G_i by a peer p' possessing $<R_i,V'>$ and requesting for $<R_i,V''>$. The algorithm for intra-group data lookup is presented in Figure 3, *Algorithm-Intra*.

```
Algorithm-Intra

  1. $p'$ broadcasts its request in $G_i$ for $<R_i, V''>$
     //one hop communication since diameter of $G_i$ is one

  2. if $\exists$ $p''$ with $<R_i, V''>$ then
           $p''$ unicasts $<R_i, V''>$ to p'
     else
           search for $<R_i, V''>$ fails
           //search latency is minimum, i.e., only two hops
```

Figure 3: Intra-group data lookup algorithm

4.2 Inter-Group Data Lookup

In our proposed architecture, any communication between a node p_i' ϵ G_i and p_j' ϵ G_j takes place only via the respective group-heads P_i and P_j. Without any loss of generality let a peer p_i' (ϵ G_i) request for $<R_j,V^*>$. The following steps are executed to answer the query. Peer p_i' knows that $R_j \notin G_i$. Assume that there are n distinct resource types. In order to locate resource R_j, a search along the transit ring network is required. The algorithm for inter-group data lookup is presented in Figure 4, *Algorithm-Inter*.

However, in our proposed architecture, number of peers on the ring is the number of distinct resource types *n* and it has been observed that the number of peers in most P2P networks is too large compared to the number of distinct resource types i.e., $n<<N$. ,Therefore, such search on the ring in our proposed architecture appears to be quite practical.

5 Concurrent Leaves and Joins

We assume that a well-known server keeps a copy of the GRT. When a new node (peer) wishes to join the system, it contacts the server. If the request to join is for an existing resource type, say R_i, the server sends the IP address of the group-head P_i to the node. If the request is for a new resource type, the server sends the IP address of the group-head P_0. Therefore, in our design the server plays a small but very important role related to load sharing by group-heads. All that is needed is when the GRT is updated by the group-heads, a copy is sent to the server. By virtue of its construction, the GRT remains sorted by default and in an ascending order of the Group-heads' logical addresses; so determining the exact group-head is $O(\log n)$.

Churn is frequent arrivals and departures of the peers in the system. Let us now see the possible scenarios of peers joining and leaving.

5.1 Concurrent Joins

As pointed out earlier, a peer p either can join an existing group, or can form a new group with the group-head being the peer itself.

Scenario 1: Peer joining an existing resource type is explained in Figure 5.

- Since nodes in a group are directly connected to each other, hence joining a group means forming a logical link between the peer p and each node in the group.
- If multiple peers join the same group, say G_k, the join requests are queued at the group head P_i and are served on FCFS basis.
- Observe that joining multiple groups can take place concurrently, because joining one group is unrelated to joining other groups.

Scenario 2: New peer with new resource type

Consider a P2P interest-based system which has S distinct resource types, viz., R_0, R_1, R_2, … R_{s-1}. Without any loss of generality, let us assume a scenario where a new peer *p* wants a data insertion for a new resource type R_s.

- In this case, it is a new resource type R_s, the joining peer

```
Algorithm-Inter

  1. $p_i'$ sends a data lookup request for $<R_j,V^*>$ to its group-head $P_i$

     // one hop communication

  2. if $P_i$ is also the group-head ($P_j$) for resource type $R_j$

        if    $P_i$ (as $P_j$) possesses $< R_j, V^*>$ then
              $P_i$ (as $P_j$) unicasts $< R_j, V^*>$ to $p_i'$
        else
              $P_i$ (as $P_j$) executes Algorithm-Intra in $G_j$
     else
3.   $P_i$ determines the group-head $P_j$'s address code from GRT

     / address code of $P_j$ = resource code of $R_j = j$

  4. $P_i$ computes $|i - j| = h$
  5.    if $h > n / 2$ then

     $P_i$ forwards the request along with the IP
     address of $p_i'$ to its immediate predecessor $P_{i-1}$

        else
      $P_i$ forwards the request along with the IP
      address of $p_i'$ to its immediate successor $P_{i+1}$

       / Looking for minimum no. of hops along the
              transit ring network
     end

  6. if an intermediate group-head $P_k$ is also the group-head for resource type Rj then

         if   $P_k$ (as $P_j$) possess $< R_j, V^*>$ then
              $P_k$ (as $P_j$) unicasts $< R_j, V^*>$ to $p_i'$
        else
              $P_k$ (as $P_j$) executes Algorithm-Intra in $G_j$ as the group head $P_j$

    else
  7. Each intermediate group-head $P_k$ forwards the request until $P_k = P_j$

            / no. of hops along the ring in worst case is $n / 2$

  8.    if    $P_j$ possess $<R_j,V^*>$  then
             $P_j$ unicasts $<R_j,V^*>$  to $p_i'$
        else
              $P_j$ executes Algorithm-Intra in $G_j$

    end
```

Figure 4: Inter-group lookup algorithm

Algorithm- Join Existing

```
1. New peer p with resource type $R_k$ unicasts its join request to $P_0$
2. $P_0$ determines the group $G_k$ for $p$ from its GRT
3. $P_0$ unicasts IP address of $p$ to $P_k$
4. $P_k$ assigns $p$ with the next available address $(k+jn)$
5. $P_k$ includes $p$ in its list of neighbors in $G_k$
6. $P_k$ asks all members of $G_k$ to include $p$ in their lists
7. $P_k$ sends the updated list of neighbors in $G_k$ to $p$
8. $p$ establishes direct logical link to all members of $G_k$
```

Figure 5: Peer joining existing resource type algorithm

contacts P_0 which is the group-head of the very first group formed in the system
- P_0 assigns a logical address *s (mod n)* of the residue system S_n to *p*.
- *p* becomes the group head P_s of new group G_s
- GRT and all the predecessor and successor pointers of group heads are updated accordingly
- Multiple such requests eventually arrive at P_0 and P_0 serves the requests on FCFS basis
- We can handle insertion of multiple new resource types by the same peer in a similar way. Note that in the proposed architecture joining of any new resource type always takes place between the recent and the first groups. This feature makes such joining localized to a single position on the ring; thereby making the joining process much simpler.

It is obvious that the above mentioned Scenario 1 and Scenario 2 can occur simultaneously.

5.2 Concurrent Leaves

It is assumed that any two directly connected peers in a group or along the transit ring periodically exchange periodic hello packets. Leaving a network could be graceful or abrupt (unexpected crash). In both the cases, if a hello packet is not received from a neighbor peer it is interpreted that the peer is no more alive(unreachable). Both the cases of a peer leaving Figure 5: Peer Joining Existing Resource type algorithm (graceful or abrupt) are handled in the same way.

Scenario 1: Group Member Crashes or Leaves

- The logical link information (i.e., logical address) of the peer left is deleted from the routing table of each peer not receiving the hello packet.
- Therefore, concurrent such leavings whether taking place in the same group or in multiple groups amounts to the deletion of the corresponding link information in the routing tables of the concerned non-leaving peers only.

Scenario 2: Group Head Crashes or Leaves

- The procedure to handle the case of a group-head crashing or leaving the network can be achieved easily with a small overhead of saving pointer values present in a group-head P_i in a peer $p^* \in G_i$.
- An update from P_i to p_i^* is triggered whenever P_i detects a change in the transit network.
- In order to guard against any loss of information due to group-head P_i's crash/leave, P_i also sends a snapshot of its request queue to p^*each time the content of the queue is updated.

Note that handling of single group-head crash has been discussed in [12]. Multiple group heads' leaving is considered in the following section.

5.3 Concurrent Joins and Leaves

Observe that 'concurrent joins and leaves' means that addition and deletion of logical links taking place concurrently.

- If a peer is involved in both actions, it will do so sequentially on FCFS basis
- Otherwise, different peers can execute these two operations concurrently in the system.

6 Comparison

6.1 Data Lookup Complexity

In Chord [15] search along the chord is not followed, because it is very inefficient in a large peer to peer system since the mean number of hops required per search is N / 2, where N is the total number of peers in the system. In our work, the mean number of hops required (on the ring network) per search is n / 2, where n is the number of distinct resources. Fact is, in general, the total number of peers N is much larger than the number of distinct resource types n; hence search along the transit ring network in our work can be quite efficient.

It is also apparent from the fact that in Chord [15] and in other structured P-2-P systems [13, 20] the complexity involved in data lookup is a function of the number of peers N in the system; where as in the proposed architecture it is a function of the number of distinct resource types n. The point to mention is that use of the same code to denote a resource type R_i and the corresponding group-head P_i has made the search process simple and efficient. Thus, the time complexity for data lookup in our presented architecture is bounded by $\left(1+\frac{n}{2}\right)$. In Table 1, we have presented data lookup complexity of our approach as well as those of some important existing DHT based systems. The look up performance comparison is represented graphically on Figure 6. Observe that from the viewpoint of data lookup complexity, our proposed architecture offers better performance.

Table 1: Data lookup complexity comparison

	Chord	**Our Work**	**Our Work**
Architecture	DHT-based	RC-based	RC-based
Lookup Protocol	Matching key and NodeID.	*Inter-Group:* Routing through Group-heads	*Intra-group:* Complete Graph
Parameters	*N*-number of peers in network.	*n* - Number of distinct resource types, N-number of peers in network, *n* << *N*	*n* - Number of distinct resource types, N-number of peers in network, *n* << *N*
Lookup Performance	*O*(log N)	Inter-Group: *O*(n)	Intra-group: *O*(1)

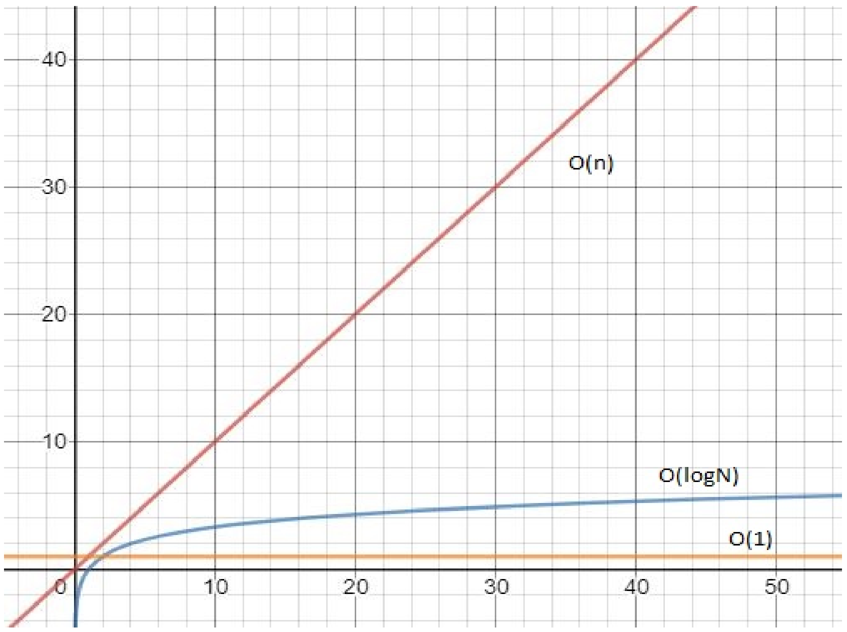


Figure 6: Lookup performance comparison

7 Conclusion

In this paper, we have extended our non-DHT based structured P2P architecture to incorporate the generic idea that a peer can possess multiple resource types. We have applied some property of modular arithmetic, specifically residue class (RC), to design a scalable, hierarchical structured overlay P2P system, which provides highly efficient data lookup algorithms. One noteworthy point is that complexity involved in data lookup is a function of the number of distinct resource types n unlike in DHT-based systems. Another point to mention is that use of the same code to denote a resource type R_i and the corresponding group-head P_i has made the search process simple and efficient. This work is a part of an ongoing research project with the goal of designing P2P federation consisting of small P2P systems so that bandwidth cannot be an issue.

References

[1] C. K. S. Banerjee and B. Bhattacharjee, "Scalable Application Layer Multicast," *Proc. ACM SIGCOMM'02*, pp. 205-217, Aug. 2002.

[2] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-like P2P Systems Scalable," *Proc. ACM SIGCOMM*, Karlsruhe, Germany, pp. 407-418, August 25-29 2003.

[3] E. Cohen, A. Fiat, and H. Kaplan, "Associative Search in Peer-to-Peer Networks: Harnessing Latent Semantics," 2:1261-1271, 2003.

[4] P. Ganesan, Q. Sun, and H. Garcia-Molina, "Yappers: A Peer-to-Peer Lookup Service over Arbitrary Topology," *Proc. IEEE Infocom 2003*, San Francisco, USA, 2:1250-1260, March 30 - April 1 2003.

[5] Bidyut Gupta, Nick Rahimi, Henry Hexmoor, Shahram Rahimi, Koushik Maddali, and Gongzhu Hu, "Design of Very Efficient Lookup Algorithms for a Low Diameter Hierarchical Structured Peer-to-Peer Network," *Proc. IEEE 16th Int. Conf. Industrial Informatics (IEEE INDIN)*, Porto, Portugal, pp. 861-868, July 2018.

[6] Swathi Kaluvakuri, Koushik Maddali, Bidyut Gupta and Narayan Debnath, "Design of RC Based Low Diameter Hierarchical Structured P2P Network Architecture," *EMENA-ISTL*, 2019; LAIS 7 (Learning and Analytics in Intelligent Systems), Springer, 7:312-320, 2020.

[7] M. Kleis, E. K. Lua, and X. Zhou, "Hierarchical Peer-to-Peer Networks using Lightweight SuperPeer Topologies," *Proc. IEEE Symp. Computers and Communications*, pp. 143-148, 2005.

[8] D. Korzun and A. Gurtov, "Hierarchical Architectures in Structured Peer-to-Peer Overlay Networks," *Peer-to-Peer Networking and Applications*, Springer, pp. 1-37, March 2013.

[9] Koushik Maddali, Banafsheh Rekabdar, Swathi Kaluvakuri and Bidyut Gupta, "Efficient Capacity-Constrained Multicastin RC based P2P Networks," *EPiC Series in Computing*, CAINE, 63:121-129, September 2019

[10] Andrea Passarella, "A Survey on Content-Centric Technologies for the Current Internet: CDN and P2P Solutions," *Computer Communications*, 35:1-32, 2012.

[11] Z. Peng, Z. Duan, J. Jun Qi, Y. Cao, and E. Lv., "HP2P: A Hybrid Hierarchical P2P Network," *Proc. Intl. Conf. Digital Society*, pp. 18-28, 2007.

[12] N. Rahimi, K. Sinha, B. Gupta, and S. Rahimi, "LDEPTH: A Low Diameter Hierarchical P2P Network Architecture," *Proc. 2016 IEEE Int. Conf. on Industrial Informatics (INDIN 2016)*, Poitiers, France, pp. 832-837, July, 2016.

[13] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large Scale Peer-to-Peer Systems," *Proc. FIP/ACM Intl. Conf. Distributed Systems Platforms (Middleware)*, pp. 329-350, 2001.

[14] K. Shuang, P Zhang, and S. Su, "Comb: A Resilient and Efficient Two-Hop Lookup Service for Distributed Communication System," *Security and Communication Networks*, 8(10):1890-1903, 2015.

[15] R. I. Stocia, R. Morris, D. Liben-Nowell, D. R. Karger, M. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," *IEEE/ACM Tran. Networking*, 11(1):17-32, Feb. 2003.

[16] M. Xu, S. Zhou, and J. Guan, "A New and Effective Hierarchical Overlay Structure for Peer-to-Peer Networks," *Computer Communications*, 34:862-874, 2011.

[17] M. Yang and Y. Yang, "An Efficient Hybrid Peer-to-Peer System for Distributed Data Sharing," *IEEE Trans. Computers*, 59(9)1158-1171, Sep. 2010.

[18] R. Zhang and Y. C. Hu, "Borg: A Hybrid Protocol for Scalable Application-Level Multicast in Peer-to-Peer Networks," *Proc. Int'l. Workshop Network and Operating System Support for Digital Audio and Video (NOSSDAV'03)*, pp. 172-179, 2003.

[19] R. Zhang and Y.C. Hu, "Assisted Peer–to-Peer Search with Partial Indexing," *IEEE Trans. Parallel and Distributed Systems*, 18(8):1146-1158, 2007.

[20] B. Y. Zhao, L. Huang, S. C. Rhea, J. Stribling, A. Zoseph, and J. D. Kubiatowicz, "Tapestry: A Global-Scale Overlay for Rapid Service Deployment," *IEEE J-SAC*, 22(1):41-53, Jan. 2004.

Swathi Kaluvakuri (photo not available) is a Ph.D. candidate from Southern Illinois University Carbondale – School of Computing. She graduated from Jawaharlal Nehru Technological Unversity with a Bachelor of Technology degree in Computer Science major. She holds a keen interest in the areas of Peer to Peer Networking and BlockChain and worked as a Software Engineer, Technical Product Support and IBM AS400 developer for NetCracker Pvt Ltd from 2012-2014.

Koushik Maddali (photo not available) is a Ph.D. candidate in Department of Computer Science at Southern Illinois University Carbondale. He received his MS from the same university and his BS from Jawaharlal Nehru Technological University, India. His research interests include Peer to Peer Networking, BlockChain and worked on a Virtual Terminal project of Cisco from 2017-2018.

Nick Rahimi (photo not available) is a Cybersecurity assistant professor in Southern Illinois University (SIU). His research interest lies in the area of Blockchain, cryptography, peer-to-peer networks, software security, and Internet censorship. He earned his Ph.D. and M.S. in Computer Science from Southern Illinois University. Nick obtained two B.S. degrees in Computer Software Engineering and Information Systems and Technologies.

Bidyut Gupta (photo not available) received his M. Tech. degree in Electronics Engineering and Ph.D. degree in Computer Science from Calcutta University, Calcutta, India. At present, he is a professor at the School of Computing (formerly Computer Science Department), Southern Illinois University, Carbondale, Illinois, USA. His current research interest includes design of architecture and communication protocols for structured peer-to-peer overlay networks, security in overlay networks, and block chain. He is a senior member of IEEE and ISCA.

Narayan Debnath (photo not available) earned a Doctor of Science (D.Sc.) degree in Computer Science and also a Doctor of Philosophy (Ph.D.) degree in Physics. Narayan C. Debnath is currently the Founding Dean of the School of Computing and Information Technology at Eastern International University, Vietnam. He is also serving as the Head of the Department of Software Engineering at Eastern International University, Vietnam. Dr. Debnath has been the Director of the International Society for Computers and their Applications (ISCA) since 2014. Formerly, Dr. Debnath served as a Full Professor of Computer Science at Winona State University, Minnesota, USA for 28 years (1989-2017). Dr. Debnath has been an active member of the ACM, IEEE Computer Society, Arab Computer Society, and a senior member of the ISCA.

A Detailed Comparison of the Effects of Code Refactoring Techniques in Different Mobile Applications

Osama Barack[*] and LiGuo Huang[†]
Southern Methodist University, Dallas, TX 75205, USA

Abstract

Due to the high usage of mobile applications among end users, developers are required to maintain and extend mobile application code. Fowler Martin introduces refactoring techniques to make software code readable, understandable, extensible and more efficient. When refactoring techniques are applied to mobile application code, they affect the energy efficiency and performance of mobile applications. In our previous study, we implemented Fowler's sample code into a mobile application and used Greenup, Powerup, and Speedup (GPS-UP) metrics to evaluate and categorize the impact of refactoring techniques. However, Fowler's sample code is simple and does not reflect an accurate evaluation of the refactoring techniques. Thus, we extend our work through presenting a case study that evaluates and categorizes the impact of refactoring techniques when they are applied to open-source mobile applications. In addition, we provide a comparison of the effect of refactoring techniques between the results of this study and our previous one.

Key Words: Energy, GPS-UP metrics, mobile application, open-source, performance, refactoring technique.

1 Introduction

The popularity of mobile devices increases the daily use of mobile applications, which leads to shortened battery life. In addition, software developers increase the complexity and features of mobile applications, which makes the software code complicated and inefficient. Software quality is concerned with measuring and addressing the quality level of software code [13], and maintainability is considered one of the main quality attributes. Therefore, Fowler Martin introduces code refactoring techniques to increase the quality of the software code. Fowler defined refactoring as "the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure" [8]. Refactoring techniques do improve the quality of the software code; however, they also affect the energy consumption and performance of the mobile applications.

Abdulsalam et al. [1] proposed Greenup, Powerup, and Speedup (GPS-UP) metrics to show the interrelationships among energy, performance, and power. GPS-UP metrics show the correlation between energy consumption and performance. In our previous work [3], we modified GPS-UP metrics by adding two categories to the original eight categories and presented a study to evaluate and categorize the impact of 21 refactoring techniques implemented Fowler's sample code in mobile environments using the modified GPS-UP metrics to make developers and programmers aware of the changing in energy efficiency and performance of mobile applications after applying refactoring techniques.

In this study, we extend our work through evaluating the impact of refactoring techniques that are applied to the software code of mobile applications that contain common algorithms (quick sort and binary search), and data structures (linked list). In addition, we evaluate the influence of refactoring techniques on open-source mobile applications (Simple Calculator and AnotherMonitor). Moreover, we compare the results of this study and the results of Fowler's sample code.

The remainder of this journal is structured as follows: Section 2 provides related work. Section 3 explains the methodology of our approach. Section 4 presents the preparation, setup, and execution. Section 5 presents the case studies. Section 6 provides a discussion and analysis of the experiment results. Finally, section 7 provides conclusions and suggestions for future work that can be achieved in this field.

2 Related Work

Software engineering is the approach of studying the design, development, testing, and maintenance of software. It ensures that software is built correctly while satisfying all requirements [4] [5] [19]. Researchers present different methodologies to overcome the limitation of mobile devices. The literature presents different approaches to enhance the performance and energy efficiency of mobile applications.

Ramirez et al. [20] presents a study that measures the energy consumption of multithreading android applications executing only Java application and compared it to the Android executing complex part of code in C programming language using Java Native Interface (JNI). The study results help developers find the

[*]Department of Computer Science. Email: obarack@smu.edu
[†]Department of Computer Science. Email: lghuang@smu.edu.

cause behind increasing mobile application energy consumption and improve application development strategies to increase energy efficiency.

Implementing refactoring techniques improves the understandability, maintainability, and extensibility of the software code. However, the impact of energy efficiency of each refactoring technique is not shown in the automated support of refactoring in IDEs. Sahin et al. [22] presents an empirical study to explore the impact of energy efficiency for 197 application with 6 refactoring techniques. Their experiment results showed the refactoring impact on energy consumption and the capability of increasing and decreasing energy consumption. In addition, the authors gave metrics that correlated with energy consumption to predict the impact of implementing refactoring techniques.

Morales et al. [18] proposes an energy-aware refactoring approach for mobile apps (EARMO), a novel anti-pattern, accounts energy consumption when refactoring mobile anti-patterns. The authors analyze the impact of eight types of anti-patterns on a testbed of 20 android applications. EARMO has been evaluated by testing three multiobjective search-based algorithms. Their experiment results show that EARMO can generate refactoring recommendations in less than a minute and remove a median 84% of anti-patterns. In addition, EARMO extend the battery life of a mobile phone by up to 29 minutes, and 68% of EARMO refactoring suggestions were found relevant by developers.

Zecena et al. [25] explores and analyzes three parallelized sorting algorithms (Odd-Even Sort, ShellSort, and QuickSort) by executing them on multicore computers. The results showed that better algorithm performance leads to more energy savings.

Rashid et al. [21] analyzes the energy consumption of different implementations of sorting algorithms in different programming languages. The experiment results showed that different combinations of algorithms and programming languages change the level of energy efficiency. The authors' study provides the basic information of selecting algorithms and identifying main factors affecting energy consumption.

Code obfuscation prevents code piracy; however, code obfuscation has become an important concern about its impact on energy efficiency on mobile application. Therefore, Sahin et al. [23] presents an empirical study on the impact of 18 code obfuscations on energy consumption. The authors' experiment included 15 usage scenarios on 11 Android applications. The experiment results indicate that using code obfuscation is likely to increase energy consumption.

Hunt et al. [12] proposed using a lock-free data structure to improve performance, scalability, and energy efficiency. Three different types of lock-free and locking data structures were implemented to run excessive workloads and compare the execution time and the energy efficiency of each data structure type. Using threads to access a shared data structure requires synchronization of the threads and assurance of the data's consistency and integrity. However, thread synchronization causes performance problems in multithreaded programs. As a result, the lock-free data showed better performance and higher energy efficiency.

3 Methodology

The main objective of this study is to profile the positive and negative impact of refactoring techniques on mobile application code using GPS-UP metrics to determine whether the impact on performance and energy consumption caused by refactoring is beneficial. Previous studies measured performance or energy efficiency without finding the interrelationship between them. The goal of finding the correlation between performance and energy efficiency is to find the cause of increasing or decreasing energy consumption when refactoring techniques are implemented. The impact of refactoring techniques on mobile application code has rarely been investigated, and previous experiments ended up just measuring performance or energy consumption for desktop applications. Therefore, this study focuses on assessing the positive or negative impact of refactoring on mobile application code. Our approach quantitatively evaluates and categorizes refactoring techniques when applied to software applications in mobile environments.

4 Experiment Preparation, Setup, and Execution

Two versions of the application software code (non-refactored and refactored) are compared to each code refactoring technique by using the modified GPS-UP metrics in order to show the improvement or decline in performance and energy efficiency. The following is a description of the experiment preparation, including a list of the selected refactoring techniques and the tools that are used for the experiment.

4.1 Selection of Code Refactoring Techniques

In this study, we evaluate and categorize the impact of 10 refactoring techniques for energy consumption and performance. Table 1 is a list of the selected code refactoring techniques organized in the order they were introduced in Fowler's book:

Table 1: The selected refactoring techniques

Group	Refactoring Technique
Composing Methods	Extract Method Inline Method Inline Temp Replace Temp with Query Split Temporary Variable
Moving Features Between Objects	Move Method
Organizing Data	Replace Array with Object
Simplifying Conditional Expressions	Move Method
Making Method Calls Simpler	Add Parameter Remove Parameter

Table 2: GPS-UP metrics categories for 10 code refactoring techniques for common algorithms code in Samsung Galaxy S5, where (s) is second and (j) is joule

Samsung Galaxy S5 - Common Algorithms	AVG of 10 Runs		AVG of 10 Runs		GPS-UP Metrics				LOC	
Code refactoring techniques	TBR(s)	EBR(j)	TAR(s)	EAR(j)	Greenup	Speedup	Powerup	Category	Total	Changed
Inline Method	88.49	28.86	86.19	27.16	1.06	1.03	0.97	C1	310	8
Move Method	89.01	29.35	86.52	28.17	1.04	1.03	0.99	C1	310	20
Remove Parameter	89.07	28.82	88.28	28.53	1.01	1.01	1.00	C2	310	8
Inline Temp	88.88	28.77	87.52	28.48	1.01	1.02	1.01	C4	310	5
Replace Array with Object	88.66	27.99	87.26	29.84	0.94	1.02	1.08	C6	310	25
Decompose Conditional	88.43	29.07	89.48	29.35	0.99	0.99	1.00	C8	310	10
Replace Temp with Query	88.57	28.30	88.79	29.61	0.96	1.00	1.04	C9	310	11
Split Temporary Variable	88.38	28.39	88.48	29.67	0.96	1.00	1.04	C9	310	5
Extract Method	87.54	27.74	88.82	29.09	0.95	0.99	1.03	C10	310	9
Add Parameter	87.32	28.36	88.78	29.20	0.97	0.98	1.01	C10	310	6

Table 3: GPS-UP metrics categories for 10 code refactoring techniques for common algorithms code in LG Nexus 5X, where (s) is second and (j) is joule

LG Nexus 5X - Common Algorithms	AVG of 10 Runs		AVG of 10 Runs		GPS-UP Metrics				LOC	
Code refactoring techniques	TBR(s)	EBR(j)	TAR(s)	EAR(j)	Greenup	Speedup	Powerup	Category	Total	Changed
Inline Method	89.17	29.02	85.80	26.13	1.11	1.04	0.94	C1	310	8
Move Method	88.34	29.31	86.99	27.45	1.07	1.02	0.95	C1	310	20
Remove Parameter	88.86	29.02	87.86	28.83	1.01	1.01	1.00	C2	310	8
Inline Temp	88.25	29.18	86.65	28.89	1.01	1.02	1.01	C4	310	5
Replace Array with Object	89.00	28.99	87.09	30.19	0.96	1.02	1.06	C6	310	25
Decompose Conditional	88.80	28.87	89.76	29.04	0.99	0.99	1.00	C8	310	10
Replace Temp with Query	88.79	29.05	88.95	31.03	0.94	1.00	1.07	C9	310	11
Split Temporary Variable	88.71	29.63	88.84	30.84	0.96	1.00	1.04	C9	310	5
Extract Method	88.87	28.85	91.05	30.79	0.94	0.98	1.04	C10	310	9
Add Parameter	88.92	28.42	90.29	30.88	0.92	0.98	1.07	C10	310	6

4.2 Experiment Tools

We use modified GPS-UP metrics to evaluate and categorize the selected refactoring techniques. The execution time of the mobile application (T) in second(s) and the energy consumption of the mobile application in joule(s) are measured to calculate Speedup and Greenup. Speedup indicates the ratio of the non-refactored code runtime to the refactored code runtime. Greenup indicates the ratio of the total energy consumption of the non-refactored code to the total energy consumption of the refactored code.

Android Studio was used to implement refactoring techniques to the mobile applications. The applications were installed on the same mobile devices (Samsung Galaxy S5 and an LG Nexus 5X) to have a suitable comparison with our previous work. We also use the same mobile application to measure energy consumption (PowerTutor [24]).

4.3 Experiment Execution Setup

The steps of the experiment are explained in algorithm 1:

Algorithm 1: Experiment Steps

```
Result: Evaluating and categorizing code refactoring techniques
Install the mobile application through Android Studio;
count = 1;
while count <= 10 do
    Run the application on the Android mobile platform;
    Measure the energy consumption (joule) and performance (speed in second);
end
Calculate the average of the energy consumption and performance;
Apply the code refactoring technique to the code;
count = 1;
while count <= 10 do
    Run the application on the Android mobile platform;
    Measure the energy consumption (joule) and performance (speed in second);
end
Calculate the average of the energy consumption and performance;
Apply the GPS-UP metrics to the experiment results;
```

Table 4: GPS-UP metrics categories for 6 code refactoring techniques for AnotherMonitor application in Samsung Galaxy S5, where (s) is second and (j) is joule

Samsung Galaxy S5 - AnotherMonitor	AVG of 10 Runs		AVG of 10 Runs		GPS-UP Metrics				LOC	
Code refactoring techniques	TBR(s)	EBR(j)	TAR(s)	EAR(j)	Greenup	Speedup	Powerup	Category	Total	Changed
Inline Method	87.54	17.42	84.47	15.53	1.12	1.04	0.92	C1	2394	70
Inline Temp	87.57	17.44	86.55	17.29	1.01	1.01	1.00	C2	2394	15
Replace Array with Object	87.56	17.49	85.52	18.49	0.95	1.02	1.08	C6	2394	60
Extract Method	87.59	17.44	87.78	18.55	0.94	1.00	1.06	C9	2394	75
Decompose Conditional	87.32	17.47	88.96	18.47	0.95	0.98	1.04	C10	2394	60
Split Temporary Variable	87.47	17.52	88.57	18.44	0.95	0.99	1.04	C10	2394	25

Table 5: GPS-UP metrics categories for 6 code refactoring techniques for AnotherMonitor application in LG Nexus 5X, where (s) is second and (j) is joule

LG Nexus 5X - AnotherMonitor	AVG of 10 Runs		AVG of 10 Runs		GPS-UP Metrics				LOC	
Code refactoring techniques	TBR(s)	EBR(j)	TAR(s)	EAR(j)	Greenup	Speedup	Powerup	Category	Total	Changed
Inline Method	89.14	18.45	85.51	17.68	1.04	1.04	1.00	C2	2394	70
Inline Temp	89.45	18.55	85.68	18.35	1.01	1.04	1.03	C4	2394	15
Replace Array with Object	87.43	19.09	89.29	19.31	0.99	0.98	0.99	C7	2394	60
Extract Method	88.87	18.64	88.91	18.74	0.99	1.00	1.01	C9	2394	75
Decompose Conditional	88.50	18.30	89.59	19.10	0.96	0.99	1.03	C10	2394	60
Split Temporary Variable	87.93	17.98	89.14	18.79	0.96	0.99	1.03	C10	2394	25

5 Case Studies

In this study, three different mobile applications are used to evaluate code refactoring techniques, common algorithms code, AnotherMonitor, and Simple Calculator. The energy consumption and execution time of the mobile application were measured ten times before implementing each refactoring technique, and ten times after implementing each refactoring technique. After the average, median, and variance of each ten runs were calculated and analyzed, there were no extreme scores, and the average value was found as the best value to be applied to the GPS-UP metrics. Then, we calculated Greenup, Speedup, and Powerup to categorize each refactoring technique.

5.1 Common Algorithms Code

In this experiment, we implement a mobile application with a code that has a common data structure (linked list) [11] and two algorithms (quick sort and binary search) in Java code [6] [21] [25]. The linked list contains 4,500 objects. Ten code refactoring techniques are applicable to the application code. After we implemented each refactoring technique to the code, results were applied to the GPS-UP metrics to categorize each refactoring technique. Table 2 and Table 3 illustrate the results of the Samsung Galaxy S5 and LG Nexus 5X, respectively.

5.2 Open-Source Applications

To generalize our experiment results, we chose two commonly used applications from F-Droid [15] software repository to measure the impact of 6 refactoring techniques on energy consumption and performance in a real open-source mobile application as in [2] [7] [10] [16] [17].

AnotherMonitor [9] is a mobile application that monitors and records CPU utilization and memory usage. It generates graphic results in 0.5, 1, 2 and 4 second intervals. To perform the experiment and measure the application energy consumption and performance, the interval time was disabled and swapped with a loop that has 20k iterations. Six code refactoring techniques out of the 21 were applicable to the mobile application code. After implementing each refactoring technique to the code, the results were applied to the GPS-UP metrics to categorize the used refactoring technique. Table 4 and Table 5 illustrate the results of the Samsung Galaxy S5 and LG Nexus 5X, respectively.

Simple Calculator [14] is a mobile application that performs simple mathematical functions. Decimal numbers were injected as an input to the application code, to eliminate the human factor and measure the application energy consumption and performance. Six code refactoring techniques out of the 21 were applicable to the mobile application code. After we implemented each refactoring technique to the code, the results were applied to the GPS-UP metrics to categorize each refactoring technique. Table 6 and Table 7 illustrate the results of the Samsung Galaxy S5 and LG Nexus 5X, respectively.

6 Discussion

The following is an explanation of the technical reasons behind the positive or negative improvement for each refactoring technique shown in the result of our case studies.

Table 6: GPS-UP metrics categories for 6 code refactoring techniques for Simple Calculator application in Samsung Galaxy S5, where (s) is second and (j) is joule

Samsung Galaxy S5 - Simple Calculator	AVG of 10 Runs		AVG of 10 Runs		GPS-UP Metrics				LOC	
Code refactoring techniques	TBR(s)	EBR(j)	TAR(s)	EAR(j)	Greenup	Speedup	Powerup	Category	Total	Changed
Inline Method	62.21	11.00	61.52	9.98	1.10	1.01	0.92	C1	700	21
Inline Temp	62.34	10.56	61.52	10.43	1.01	1.01	1.00	C2	700	10
Replace Array with Object	61.78	10.33	60.16	11.00	0.94	1.03	1.09	C6	700	15
Extract Method	62.82	10.39	62.99	11.28	0.92	1.00	1.08	C9	700	35
Decompose Conditional	62.76	11.81	63.25	12.11	0.98	0.99	1.02	C10	700	29
Split Temporary Variable	61.16	11.32	62.33	12.02	0.94	0.98	1.04	C10	700	17

Table 7: GPS-UP metrics categories for 6 code refactoring techniques for Simple Calculator application in LG Nexus 5X, where (s) is second and (j) is joule

LG Nexus 5X - Simple Calculator	AVG of 10 Runs		AVG of 10 Runs		GPS-UP Metrics				LOC	
Code refactoring techniques	TBR(s)	EBR(j)	TAR(s)	EAR(j)	Greenup	Speedup	Powerup	Category	Total	Changed
Inline Method	63.67	10.37	62.28	10.10	1.03	1.02	1.00	C2	700	21
Inline Temp	63.04	10.17	61.71	10.03	1.01	1.02	1.01	C4	700	10
Replace Array with Object	62.99	10.91	64.13	10.99	0.99	0.98	0.99	C7	700	15
Extract Method	62.75	10.03	62.86	11.28	0.89	1.00	1.12	C9	700	35
Decompose Conditional	63.21	11.09	63.68	11.82	0.94	0.99	1.06	C10	700	29
Split Temporary Variable	63.22	11.26	63.84	13.11	0.86	0.99	1.15	C10	700	17

6.1 Green Categories

The refactoring techniques that fell in the green area of GPS-UP metrics improved performance or energy efficiency or both together. The Inline Method technique replaced the method call with its body which eliminates the fetch-decode-execute cycle to call the method. The Move Method technique reduced the cost of the queries between two classes by moving the method to the class that has more features with it. The Remove Parameter technique deleted the parameter that is no longer needed in the method. As a result, the refactoring technique eliminated the extra load in the memory.

The Inline Temp technique improved the performance more than the energy efficiency. Deleting temporary variables reduced the time for fetching the unnecessary temporary variable from the main and cache memories. As SpeedUp is greater than PowerUp, the Inline Temp technique is still considered a positive improvement to the energy efficiency.

6.2 Red Categories

The refactoring techniques that fell in the red area of GPS-UP metrics consumed more energy although several refactoring techniques improved performance. The Replace Array With Object technique changed the array that had different types of elements into an object and the different types of elements into the object's attributes which made the code more understandable and faster over the cost of consuming more energy for accessing the object's attributes instead of the array's elements. The Decompose Conditional technique replaced each part of a complicated condition if-then-else into a method which is more readable and understandable; however, calling the created methods costs more energy. The Extract Method technique extracted part of the long method to be in a new separate method. As a result, the refactoring technique downgraded the performance and energy efficiency for calling the new method. However, this technique is still beneficial because the new method can be called by other methods.

The Replace Temp With Query technique replaces the temporary variable and its references with a query from a method, which made the CPU execute the method at every reference to the temporary variable. The positive improvement is that the created query and its method can be used by other methods. The Split Temporary Variable technique replaced a temporary variable that is assigned to two values with two temporary variables, which makes the code more understandable. The price was sacrificing performance and energy by loading two variables instead of one to the memory.

The last refactoring technique, the Add Parameter technique, adds a parameter to the method that needed more information, which means more energy is needed to access this parameter from the RAM. These refactoring techniques do not improve performance or energy efficiency despite being useful for reaching maintainability.

6.3 Analysis and Comparison

The four case studies are compared next to each other in Table 8 based on GPS-UP metrics categories for refactoring techniques. In Fowler's Sample, the categories of the 21 refactoring techniques are slightly different in the two mobile environments. However, the 21 refactoring techniques fell within the same green or red area of the GPS-UP metrics. In common algorithms, the 10 refactoring techniques fell exactly in the same GPS-UP metrics categories in both mobile

Table 8: Comparison of GPS-UP metrics categories for code refactoring techniques in the three case studies, where (-) is not applicable

	Categories							
	Fowler's Sample		Common Algorithms		AnotherMonitor		Simple Calculator	
Code refactoring techniques	Samsung Galaxy 5S	LG Nexus 5X	Samsung Galaxy 5S	LG Nexus 5X	Samsung Galaxy 5S	LG Nexus 5X	Samsung Galaxy 5S	LG Nexus 5X
Inline Method	C1	C1	C1	C1	C1	C2	C1	C2
Move Method	C1	C1	C1	C1	-	-	-	-
Inline Class	C1	C1	-	-	-	-	-	-
Remove Parameter	C2	C1	C2	C2	-	-	-	-
Pull Up Field	C3	C1	-	-	-	-	-	-
Pull Up Method	C3	C3	-	-	-	-	-	-
Inline Temp	C4	C4	C4	C4	C2	C4	C2	C4
Remove Assignments to Parameters	C4	C5	-	-	-	-	-	-
Replace Type Code with State Strategy	C4	C5	-	-	-	-	-	-
Replace Method with Method Object	C6	C6	-	-	-	-	-	-
Move Field	C6	C6	-	-	-	-	-	-
Extract Class	C7	C6	-	-	-	-	-	-
Replace Array with Object	C6	C6	C6	C6	C6	C7	C6	C7
Decompose Conditional	C9	C9	C8	C8	C10	C10	C10	C10
Extract Method	C10	C10	C10	C10	C9	C9	C9	C9
Replace Temp with Query	C9	C10	C9	C9	-	-	-	-
Split Temporary Variable	C9	C10	C9	C9	C10	C10	C10	C10
Replace Data Value with Object	C8	C10	-	-	-	-	-	-
Self Encapsulate Field	C10	C10	-	-	-	-	-	-
Replace Conditional with Polymorphism	C10	C10	-	-	-	-	-	-
Add Parameter	C10	C10	C10	C10	-	-	-	-

environments (Samsung Galaxy 5S and LG Nexus 5X) because the code is very simple and only has one Java class. In Another Mobile and Simple Calculator, the categories of the 6 refactoring techniques fell in the same GPS-UP metrics categories in each mobile environment. However, within the same application, different mobile environments led to slightly different GPS-UP metrics categories within the same green or red area. However, in all case studies, each refactoring technique had the same impact on energy consumption and performance in each mobile application code and environment.

7 Conclusions and Future Work

This paper provides an evaluation of the impact of code refactoring techniques for energy efficiency and performance in mobile environments using GPS-UP metrics. We present a case study using GPS-UP metrics to evaluate refactoring techniques in mobile application code that contains common algorithms (quick sort and binary search), data structures (linked list), and two real open-source mobile applications (Simple Calculator and AnotherMonitor). In addition, we provide a discussion and analysis of the case studies results and explained the correlation between performance and energy efficiency for each of the chosen refactoring techniques. Moreover, we provide a comparison between the results of this study and the experiment results of Fowler's sample code. Our work helps application software developers become aware of the effects of refactoring techniques in real mobile applications.

In future work, we will evaluate the impact of additional refactoring techniques in different mobile environments to profile all refactoring techniques. In addition, we will evaluate refactoring techniques by applying different metrics other than GPS-UP metrics

References

[1] Sarah Abdulsalam, Ziliang Zong, Qijun Gu, and Meikang Qiu. Using the Greenup, Powerup, and Speedup Metrics to Evaluate Software Energy Efficiency. In *Proceedings of the 2015 Sixth International Green and Sustainable Computing Conference (IGSC)*. IEEE Computer Society, pp. 1–8, 2015.

[2] Abhijeet Banerjee, Lee Kee Chong, Sudipta Chattopadhyay, and Abhik Roychoudhury. Detecting Energy Bugs and Hotspots in Mobile Apps. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. ACM, pp. 588–598, 2014.

[3] Osama Barack and LiGuo Huang. Effectiveness of Code Refactoring Techniques for Energy Consumption in a Mobile Environment. In *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*. The Steering Committee of The World Congress in Computer Science, pp. 165–171, 2018.

[4] Bruce I. Blum. *Software Engineering: A Holistic View*. Oxford University Press, Inc., 1992.

[5] B.W. Boehm. Software Engineering. volume C-25:1226-1241, December 1976.

[6] C. Bunse, H. Höpfner, E. Mansour, and S. Roychoudhury. Exploring the Energy Consumption of Data Sorting Algorithms in Embedded and Mobile Environments. In *2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*, pp. 600–607, May 2009.

[7] Jason Flinn and Mahadev Satyanarayanan. Powerscope: A Tool for Profiling the Energy Usage of Mobile Applications. In *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA'99. Second IEEE Workshop on*. IEEE, pp. 2–10, 1999.

[8] M. Fowler. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[9] GNU. AnotherMonitor Application. `https://f-droid.org/en/packages/org.anothermonitor`, 2015.

[10] Shuai Hao, Ding Li, William GJ Halfond, and Ramesh Govindan. Estimating Mobile Application Energy Consumption Using Program Analysis. In *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, pp. 92–101, 2013.

[11] Samir Hasan, Zachary King, Munawar Hafiz, Mohammed Sayagh, Bram Adams, and Abram Hindle. Energy Profiles of Java Collections Classes. In *Proceedings of the 38th International Conference on Software Engineering*. ACM, pp. 225–236, 2016.

[12] Nicholas Hunt, Paramjit Singh Sandhu, and Luis Ceze. Characterizing the Performance and Energy Efficiency of Lock-Free Data Structures. In *Interaction between Compilers and Computer Architectures (INTERACT), 2011 15th Workshop on*. IEEE, pp. 63–70, 2011.

[13] Capers Jones. Applied Software Measurement: Global Analysis of Productivity and Quality, 2008.

[14] Tibor Kaputa. Simple calculator application. `https://f-droid.org/en/packages/com.simplemobiletools.calculator`, 2016.

[15] F-Droid Limited. F-droid. `https://www.f-droid.org`, 2010.

[16] Grace Metri, Weisong Shi, and Monica Brockmeyer. Energy-Efficiency Comparison of Mobile Platforms and Applications: A Quantitative Approach. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*. ACM, pp. 39–44, 2015.

[17] Radhika Mittal, Aman Kansal, and Ranveer Chandra. Empowering Developers to Estimate App Energy Consumption. In *Proceedings of the 18th annual international conference on Mobile computing and networking*. ACM, pp. 317–328, 2012.

[18] Rodrigo Morales, Rubén Saborido, Foutse Khomh, Francisco Chicano, and Giuliano Antoniol. EARMO: an Energy-Aware Refactoring Approach for Mobile Apps. *IEEE Transactions on Software Engineering*, (1):1–1, 2017.

[19] Peter Naur and Brian Randell. Software Engineering: Report of a Conference Sponsored by the NATO Science Committee. October 1969.

[20] R. I. Ramirez, E. H. Rubio, A. M. Viveros, and I. M. T. Hernández. Differences of Energetic Consumption Between Java and JNI Android Apps. In *2014 International Symposium on Integrated Circuits (ISIC)*, pp. 348–351, December 2014.

[21] Mohammad Rashid, Luca Ardito, and Marco Torchiano. Energy Consumption Analysis of Algorithms Implementations. In *Empirical Software Engineering and Measurement (ESEM), 2015 ACM/IEEE International Symposium on*. IEEE, pp. 1–4, 2015.

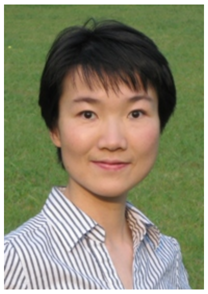
[22] Cagri Sahin, Lori Pollock, and James Clause. How Do Code Refactorings Affect Energy Usage? In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. ACM, p. 36, 2014.

[23] Cagri Sahin, Philip Tornquist, Ryan Mckenna, Zachary Pearson, and James Clause. How Does Code Obfuscation Impact Energy Usage? In *Software Maintenance and Evolution (ICSME), 2014 IEEE International Conference on*. IEEE, pp. 131–140, 2014.

[24] Z Yang. Powertutor-a Power Monitor for Android-Based Mobile Platforms. *EECS, University of Michigan, retrieved Dember*, 2:19, 2012.

[25] Ivan Zecena, Ziliang Zong, Rong Ge, Tongdan Jin, Zizhong Chen, and Meikang Qiu. Energy Consumption Analysis of Parallel Sorting Algorithms Running on Multicore Systems. In *Green Computing Conference (IGCC), 2012 International*. IEEE, pp. 1–6, 2012.

Osama Barack is a PhD candidate in the Department of Computer Science with a specialization in Software Engineering at Southern Methodist University (SMU). He also works for the Institute of Public Administration as a faculty member in Saudi Arabia. His research interests include mobile systems, virtual systems, performance, and green energy.



LiGuo Huang is an Associate Professor in the Department of Computer Science and Engineering at Southern Methodist University (SMU). Her research area is software engineering, with an emphasis on software quality/information dependability and value-based software engineering.

Journal Submission

The International Journal of Computers and Their Applications is published four times a year with the purpose of providing a forum for state-of-the-art developments and research in the theory and design of computers, as well as current innovative activities in the applications of computers. In contrast to other journals, this journal focuses on emerging computer technologies with emphasis on the applicability to real world problems. Current areas of particular interest include, but are not limited to: architecture, networks, intelligent systems, parallel and distributed computing, software and information engineering, and computer applications (e.g., engineering, medicine, business, education, etc.). All papers are subject to peer review before selection.

A. Procedure for Submission of a Technical Paper for Consideration

1. Email your manuscript to the Editor-in-Chief, Dr. Ziping Liu at: zliu@semo.edu.
2. Illustrations should be high quality (originals unnecessary).
3. Enclose a separate page (or include in the email message) the preferred author and address for correspondence. Also, please include email, telephone, and fax information should further contact be needed.
4. **Note**: Papers shorter than 10 pages long will be returned.

B. Manuscript Style:

1. The text should be **double-spaced** (12 point or larger), **single column** and **single-sided** on 8.5 X 11 inch pages.
2. An informative abstract of 100-250 words should be provided.
3. At least 5 keywords following the abstract describing the paper topics.
4. References (alphabetized by first author) should appear at the end of the paper, as follows: author(s), first initials followed by last name, title in quotation marks, periodical, volume, inclusive page numbers, month and year.
5. The figures are to be integrated in the text after referenced in the text.

C. Submission of Accepted Manuscripts

1. The final complete paper (with abstract, figures, tables, and keywords) satisfying Section B above in **MS Word format** should be submitted to the Editor-in-Chief. If one wished to use LaTex, please see the corresponding LaTex template.
2. The submission may be on a CD/DVD or as an email attachment(s). **The following electronic files should be included:**
   - Paper text (required).
   - Bios (required for each author).
   - Author Photos (jpeg files are required) or photos can be integrated into the text.
   - Figures, Tables, and Illustrations. These should be integrated into the paper text file.
3. Reminder: The authors photos and short bios should be integrated into the text at the end of the paper. All figures, tables, and illustrations should be integrated into the text.
4. The final paper should be submitted in (a) pdf AND (b) either Word or LaTex. For those authors using LaTex, please follow the guidelines and template.
5. Authors are asked to sign an ISCA copyright form (http://www.isca-hq.org/j-copyright.htm), indicating that they are transferring the copyright to ISCA or declaring the work to be government-sponsored work in the public domain. Also, letters of permission for inclusion of non-original materials are required.

Publication Charges

After a manuscript has been accepted for publication, the contact author will be invoiced a publication charge of **$500.00 USD** to cover part of the cost of publication. For ISCA members, publication charges are **$400.00 USD** publication charges are required.

Revised 2019

ISCA INTERNATIONAL JOURNAL OF COMPUTERS AND THEIR APPLICATIONS Vol. 27 No. 2, June 2020