Scalable Computing:
Practice and Experience

Scientific International Journal
for Parallel and Distributed Computing

ISSN: 1895-1767

Universitatea de Vest din Timişoara

Volume 18(4) December 2017

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

Scalable Computing: Practice and Experience

Volume 18, Number 4, December 2017

TABLE OF CONTENTS

DOI 10.12694/scpe.v18i4.1328
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 18, Number 4, pp. iii–iv. http://www.scpe.org

INTRODUCTION TO THE SPECIAL ISSUE ON COMMUNICATION, COMPUTING, AND NETWORKING IN CYBER-PHYSICAL SYSTEMS

The new paradigms and tremendous advances in computing, communications and control have provided and supported wide range of applications in all domains of live, in particular, bridging the physical components and the cyber space leading to the Cyber Physical Systems (CPS). The notion of CPS is to use recent computing, communication, and control methods to design and operate intelligent and autonomous systems that can provide using cutting edge technologies. This require the use of computing resources for sensing, processing, analysis, predicting, understanding of data, and then communication resources for interaction, intervene, and interface management, and finally provide control for systems so that they can inter-operate, evolve, and run in a stable evidence-based environment. CPS has extraordinary significance for the future of several industrial domains and hence, it is expected that the complexity in CPS will continue to increase due to the integration of cyber components with physical and industrial systems.

The special issue publish six papers. Some of them are extended from papers presented at IEEE IPCCC 2016 workshop. The first paper provides a review of cyber space and security, cyber security capability maturity models and presents security metrics framework. The work presented has several application in CPS, in particular security domain. The second paper presents a solution called "AutoReplica" for automatic and scalable data replication management in distributed computation and storage infrastructure of cyber- physical systems using SSD-HDD storage. The method presented in the paper open potential applications for CPS with distributed storage requirements. The next paper investigates the middleware challenges for CPS, based on the different types of CPS applications being developed and their specific challenges. The paper also presents developing methods for middleware platforms for CPS and shows that middleware development is relevant for several CPS applications. The fourth paper presents a design and implementation of an industrial compliant SCADA test bed using formal analysis. The method is used to differentiate attack vector by identifying influential nodes using formal concept analysis of semantics and security of Modbus/TCP protocol. The paper shows that formal methods have several potential applications in the domain of CPS. The fifth paper presents technologies, algorithms, and techniques used in smart Radio Frequency Identification systems based inventory systems. The paper differentiates the applications and capabilities of several RFID based technologies in inventory systems. Finally, the last paper present a scalable network architecture called MooreCube. The architecture allow each multi-port server directly connected to other servers via bidirectional links, without using any switch. Furthermore MooreCube is a recursively defined architecture that uses Moore graph as Building Block and uses the hierarchical structure to meet high scalability. The paper provides scalable solution with several CPS potential applications.

Amjad Gawanmeh, Department of Electrical and Computer Engineering, Khalifa University, UAE
Kashif Saleem, Centre of Excellence in Information Assurance, King Saud University, Riyadh, Saudi Arabia

DOI 10.12694/scpe.v18i4.1329
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 18, Number 4, pp. 277–290. http://www.scpe.org

CAPABILITY MATURITY MODEL AND METRICS FRAMEWORK FOR CYBER CLOUD SECURITY

NGOC T. LE, DOAN B. HOANG*

Abstract. Cyber space is affecting all areas of our life. Cloud computing is the cutting-edge technology of this cyber space and has established itself as one of the most important resources sharing technologies for future on-demand services and infrastructures that support Internet of Things (IOTs), big data platforms and software-defined systems/services. More than ever, security is vital for cloud environment. There exist several cloud security models and standards dealing with emerging cloud security threats. However, these models are mostly reactive rather than proactive and they do not provide adequate measures to assess the overall security status of a cloud system. Out of existing models, capability maturity models, which have been used by many organizations, offer a realistic approach to address these problems using management by security domains and security assessment on maturity levels. The aim of the paper is twofold: first, it provides a review of capability maturity models and security metrics; second, it proposes a cloud security capability maturity model (CSCMM) that extends existing cyber security models with a security metric framework.

Key words: Cyber security; Cloud security model; Capability Maturity Model; Security Maturity Model; Security metrics framework.

AMS subject classifications. 68M14, 68N30

1. Introduction.

In order to protect a cloud cyber space from numerous security threats, many security models and standards have been developed. Each model focuses on a particular security angle such as risk, asset, identification, physical components, network, data, and application. Few security models consider the security of a system as a whole. It is known that a single minor vulnerability can bring down the whole system and there are myriads of these vulnerabilities. Moreover, these models are inadequate because their security assessment processes are mainly about compliances and they lack meaningful and relevant quantitative security metrics.

In recent years, several security maturity models have been proposed for overall security management. These draw on the theoretical framework of the capability maturity model. In 1989, Humphrey recommended a capability maturity model for software quality assessing [1]. This basic model has been adapted for cyber security for a number of reasons. First, security models based on capability maturity model have been applied with reasonable successes for many fields such as IT, business. Second, maturity models provide a complete management process for cyber security. Third, they can be extended to cover many security aspects or domains. Recently, maturity models have been applied in securing many important traditional cyber spaces such as e-government, e-commerce, education, health, particular in critical national infrastructures such as electricity, water supply, petrol, and transportation [2]. However, few focus on cloud computing security.

Despite having the abovementioned benefits, maturity models have revealed many drawbacks. One of which is that when organizations use maturity models, they look at each maturity level as a target and build their goal to reach the next level up. The problem is that a maturity level is often determined arbitrarily and subjectively. Another issue is that security metrics mainly depend on qualitative measurements, suitable for checking compliance rather than inspiring security action.

To overcome the weaknesses and to take advantages of maturity models, we propose a capability maturity based model for cloud security, the Cloud Security Capability Maturity Model (CSCMM) with a new metrics framework that allows not only managers to assess the security state of the cloud system for decision making process but also security practitioners to identify security gaps and to implement security responses systematically and quantitatively.

The paper has several contributions:

- The paper provides a discussion on cyber security models and standards, capability maturity models, and the need for quantitative security metrics in modelling cyber security holistically to enable the

*Virtual Infrastructure and Cyber Security lab (VICS), Faculty of Engineering and IT, University of Technology Sydney (NgocThuy.Le@student.uts.edu.au, Doan.Hoang@uts.edu.au). Questions, comments, or corrections to this document may be directed to those email address.

assessment of the overall security of a complex entity.

- The paper proposes Cloud Security Capability Maturity Model (CSCMM) that embraces new cloud security domains and renders a quantitative assessment of the overall security with the system of security maturity levels. By doing so, we expand, enrich the capability maturity model theory and apply it to cloud security.
- The paper introduces a security metric framework that underpins the assessment of security maturity level. This framework supports the roadmap to create new security quantitative metrics based on the requirements of cloud stakeholders. Furthermore, it integrates previous qualitative security metrics to assess maturity level of the cloud system.

The remainder of this paper is organized as follows. Section 2 revises knowledge about cyber security, cloud security models, cyber security maturity models, and security metrics. Section 3 proposes CSCMM including its structure and implementation process. Section 4 introduces the security metrics framework that is developed to support the CSCMM model. Section 5 discusses the importance of the quantitative security metrics in security assessment process of the CSCMM model and discusses several advanced security metrics that can be applied for the model. Section 6 concludes the paper with future research.

2. Review of cyber space and security, cloud security models, security maturity models, and security metrics.

2.1. Cyber space and cyber security. The definition of cyber space has changed considerably since Wiener defined cybernetics in 1948 as control and communication in the animal and the machine [3]. Over the last few decades, academic organizations focused on the tangible elements in the cyber space when they paid more attention to the infrastructure components of IT systems, and on intangible elements such as the data or the applications within these systems. Recently, the cyber space has grown to include social networks, clouds, Internet of Things (IOTs), smart cities, smart grids, and other software-defined systems [4]. In order to provide a common understanding of the space and its security, we suggest a unified definition of the cyber space as the space that embraces all three key elements: real and virtual entities, interconnecting infrastructure, and interaction among entities. Interaction as it is fundamental to security; without interaction among entities, including human beings, the question on security may not make sense.

The definition of cyber security has evolved greatly over the past decades. The fundamental concept of security is defined as the quality or state of being secure - being free from danger [5]. Similarly, cyber security can be thought of as a system of processes that protect the resources of a cyber space. However, definitions of cyber security vary with different organizations, some using the term cyber security but others using the terms information security or IT security [6]. We highlight several definitions of cyber security for discussion and clarification. According to Gasser and Morrie [7], computer security, also known as cyber security or IT security, is the protection of information systems from theft or damage to the hardware, the software, and to the information on them, as well as from disruption or misdirection of the services they provide. International Telecommunication Union (ITU) [8] defines cyber security as the collection of tools, policies, security concepts, security safeguards, guidelines, risk management approaches, actions, training, best practices, assurance and technologies that can be used to protect the cyber environment and organization and users assets. From these definitions, it is apparent that information security emphasizes the confidentiality, integrity and availability of information whereas computer security focuses on the availability, integrity, and correct operation of systems. Furthermore, cyber space is expanding to include virtualized infrastructures, service networks, social networks, and internet of things, hence a more embracing definition is needed: cyber security can be considered as a collection of systems, tools, processes, practices, concepts and strategies that are used to prevent and protect the cyber space from unintended interaction and unauthorized access and to preserve the confidentiality, integrity, availability, authenticity, accountability (CIAAA) and other properties of the space and its resources.

This definition clarifies the scope of cyber security in three aspects. Firstly, the term cyber security is used to focus attention on security of the cyber space rather than the security in a narrower sense. Secondly, prevention, not just protection is an integral part of the definition. It makes sense to look at security in a wider context where prevention and protection are interrelated. Preventing some vulnerability from being exploited can be considered as protecting the space and on the other hand, knowing how to protect the cyber space implies to some extent the knowledge of how security breaches occur and how they can be prevented. Thirdly,

with rapid emergence of many modern technologies, such as virtualized infrastructures (cloud, software defined networks, network functions virtualization), internet of things, social networks, service networks and other new and emerging technologies, additional considerations, including adaptability, resiliency or safety should be included in the definition.

2.2. Cloud security models and standards. Cloud is a particular cyber space. Based on virtualization and shared IT resources, cloud computing is seen as a technological evolution of cyber space. It plays an important role in the world IT development and it will continue to evolve extensively over the next decades [9]. However, clouds, as cyber infrastructures, with three service models (IaaS, PaaS, and SaaS), four deployment cloud types (Private, Public, Hybrid, and Community) are facing challenging security issues.

Identified cloud security aspects include governance and compliance, virtualization, identity management [10][11][12], and various threats aspects [13][14]. Cloud Security Alliance (CSA) published the security report namely 'The Treacherous Twelve Cloud Computing Top Threats in 2016' providing organizations with the awareness of cloud security issues in making educated risk-management decisions [15].

To combat cloud security problems, researchers, businesses, and organizations have been making efforts to mitigate cloud security risk and tackle security threats by development cloud security standards and models. In 2014, the European Union Agency for Network and Information Security (ENISA) [16] released the report Cloud standards and security to provide an overview of standards relevant for cloud computing security. CSA introduced and developed security guidance for critical areas of focus in cloud computing through 3 versions including Version 1.0 [17], Version 2.1 [18] (2009), and Version 3.0 [19] (2011). The latest version (Version 3.0) was tailored for meeting the security demand changes. The aim of this guidance was to introduce better standards for organizations to manage cyber security for cloud by implementation security domains. The guidance approached cloud architecture with cloud service model (SaaS, PaaS, and IaaS) and four deployment models (Public, Private, Community, and Hybrid Cloud) with derivative variations that address specific requirements. The guidance focuses on thirteen different domains which are divided into two general categories: governance and operations. The governance domains focus on broad and strategic issues as well as policies within a cloud computing environment, while the operational domains focus on more tactical security concerns and implementation within the architecture.

This guidance is relevant to cloud computing, its service models and its deployment models. Regarding cloud security management, the guidance focuses on cloud-specific issues: interoperability and portability, data security, and virtualization. Dividing the implementation domains into two groups with strategic and tactical categories is another salient point of the guidance. This approach allows cloud consumers and providers to bring financial and human resources into security consideration. Furthermore, the guidance can be mapped to existing security models such as Cloud Control Matrix [20]. Despite these benefits, however, the guidance has a number of drawbacks. The guidance lacks assessment guide for each domain. It does not consider security metrics for security practices. Therefore, organizations find it difficult to determine the security level of a domain.

In addition, there are many standards concerning cloud security. The ISO/IEC (A joint technical committee of the International Organization for Standardization - ISO and the International Electrotechnical Commission - IEC) 27017 Standard illustrates the information security elements of cloud computing. It assists with the implementation of cloud-specific information security controls, supplementing the guidance in ISO 27000 series standards, including ISO/IEC 27018 on the privacy aspects of cloud computing, ISO/IEC 27031 on business continuity, and ISO/IEC 27036-4 on relationship management. The National Institute of Standards and Technology NIST released the following standards on cloud computing: NIST SP 500-291, Cloud Computing Standards Roadmap, NIST SP 800-146, Cloud Computing Synopsis and Recommendations, NIST SP 800-144, Guidelines on Security & Privacy in Public Cloud Computing, NIST SP 500-292, Cloud Computing Reference Architecture and NIST SP 500-293, US Cloud Computing Technology Roadmap.

2.3. Cyber Security Maturity Model. A fundamental question that has to be asked concerning a cyber space or a system is whether the cyber space or the system is secure or at least to what level it is secure. For example, is a cyber space secure when a huge number of bugs, viruses, spams and malwares have been found and fixed? Or is a cyber space secure when substantial investment in a firewall system and an IDPS (intrusion detection and prevention system) has been made? It is difficult to claim that a cyber space is safe and secure based on the numbers of vulnerabilities found and fixed as there may be a number of bugs still undetected.

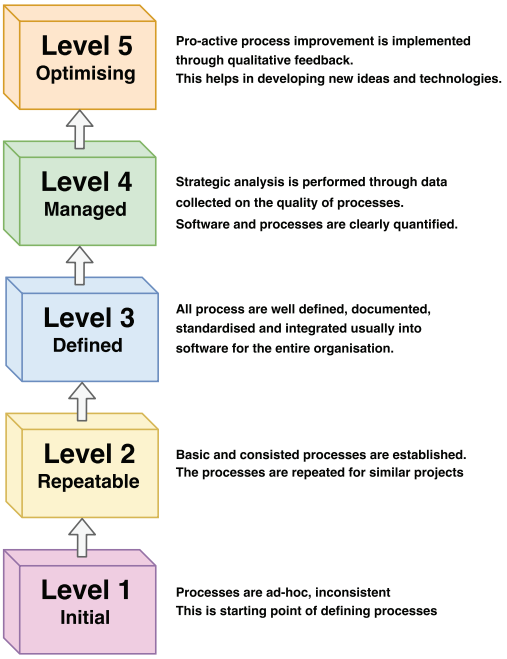


FIG. 2.1. *Capabilities maturity model process levels*

This implies that vulnerability is only one of the many aspects of security. Yet, many of the current security models deal with security problems in an ad hoc manner; a specific security measure is put into action simply to treat the issue at hand without regard to or understanding its impact on the whole cyber space. These models handle security from a bottom-up perspective and are case specific. They provide no assurance of the overall level of security of the protected entity.

What is needed is to view and study cyber security holistically from a top-down perspective to produce a security model that allows us to make an assessment of the overall security level of the entity requiring protection. Furthermore, the model should allow us to identify the entity's weaknesses and the appropriate measures to deal with them. Measures may include an investment in resources, and the enforcement of practices. Among those proposed models, the cyber-security maturity model provides organizations to some extent a roadmap for measuring, assessing, and enhancing cyber security. Relative to other models, it provides managers with sound footing for making an informed security assessment of their organization.

As mentioned above, Maturity Models are based on the Capability Maturity Model (CMM). Humphrey [1] recommended the CMM to assess quality of software and to help software organizations improve the maturity of their software processes by evolving from ad hoc, chaotic processes to mature, disciplined software processes. The fundamental ideas of CMM are as follows: (1) the model is divided into 5 levels from initial to optimizing level, from simple to complex, from low to high requirement; (2) each level has a set of maturity requirements. It means that to achieve a specific maturity level, the standard requirements of quality and technology need to be implemented by specified sets of practices; (3) to reach a higher maturity level, the software must satisfy all lower levels (Figure 2.1). Eventually, maturity models show the level of perfection or completeness of certain capabilities. They define maturity levels which measure the completeness of the analyzed objects via different sets of (multi-dimensional) criteria.

The structure of the cyber security maturity model can be described in terms of its functions, key components, and types of maturity model [21]. There are three main functions of a maturity model: a means of assessing and benchmarking performance; a roadmap for model-based improvement; and a means to identify gaps and develop improvement plans. The key components include: maturity levels which are the security measurement scales or transitional states, security domains which are logical groups of practices and processes, attributes which are core contents of the model arranged by domains and levels, diagnostic methods for assess-

ment, measurement, gap identification, benchmarking, and improvement roadmaps to guide improvement efforts such as Plan-Do-Check-Act or Observe-Orient-Decide-Act. The three types of maturity models are progression, capability, and hybrid. While the progression model describes levels as higher states of achievement similar to the progression of human mobility being from crawl, walk, jog to run, the capability model shows levels as the extent to which a particular set of practices has been institutionalized. The hybrid model is the combination of the best features of progression and capability maturity models where maturity levels express both achievement and capability. Most recent cyber security maturity models are hybrid models which describe maturity security levels over distinguished domains of a system (such as a cloud) in an integrated framework.

In our previous paper [22], we compared twelve security maturity models in order to investigate their strengths and weaknesses. It was demonstrated that cyber security maturity models help managers to manage more effectively the security of their organizations [23][24]. They allow better security risk management, produce cost saving, promotes self-improvement, and support good security procedures and processes. More importantly, they encourage all stakeholders to take steps along a secure mature path as mapped out by the maturity model, rather than activate security controls blindly without regard to the security of the overall organization. Despite all these benefits, maturity models only provide a bare minimum compliance model rather than an aspired cyber security model that can deal with emerging cyber environment, its demanding usage, as well as its sophisticated attacks. Therefore, three specific issues from security maturity models should be addressed. First, identifying the maturity levels of cyber security of each domain is arbitrary and subjective as a result of checking for compliances; a security model should be more than compliance. Second, most cyber security maturity models draw on international cyber security standards such as ISO27000 series or NIST. Security practices in these standards are mainly measured by qualitative metrics/processes; quantitative metrics should be essential for any security assessment. Third, the model should be flexible for addressing specific dimension of a cyber space or extensible for dealing with emerging cyber spaces.

2.4. Cyber security metrics.

Metrics and measures. To assess the level of a security state, metrics or measurements have been used. The usage these two terms, however, has different meanings and implications. Metrics imply tools to facilitate decision making and improve performance and accountability through collection, analysis, and reporting of relevant performance-related data. A measure is a concrete, objective attribute, such as the percentage of systems within an organization that are fully patched, the length of time between the release of a patch and its installation on a system, or the level of access to a system that a vulnerability in the system could provide. Measures are quantifiable, observable, and objective data supporting metrics [25]. According to the Information Assurance Technology Analysis Center (IATAC), a measurement is the act or the process of measuring, where the value of a quantitative variable in comparison to a (standard) unit of measurement is determined. A measure is a variable to which a value is assigned as a result of the measurement. A metric is a system of related measuring enabling quantification of some characteristic of a system, component or process. A metric is composed of two or more measures [26].

Importance of security metrics. Lord Kelvin [27] stated that when you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind. Therefore, metrics are needed to assess the security of the cyber space. In terms of software quality assessment, Humphrey [28] insisted that quality management is impossible without quality measures and quality data. As long as software people try to improve quality without measuring and managing quality, they will make little or no progress. Trapero et al. [29] indicated the importance of quantitative security metrics.

However, it is difficult to measure the cyber security state for 3 reasons: vulnerabilities are hard to measure by anyone, even the owner of the system; the set of weakness (vulnerabilities) known to the observer is not known by the owner of the system and thus is not measured by the owner; no system owner can know the totality of his adversaries. Despite having several difficulties in security measuring, cyber security metrics can support organizations in (1) verifying that their security controls are in compliance with a policy, process, or procedure, (2) identifying their security strengths and weaknesses; and (3) identifying security trends, both within and outside the organizations control [30].

Security metrics categories. Security metrics can be categorized by what and how they are measured. What are measured may include process, performance, outcomes, quality, trends, conformance to standard, and probabilities. How these things are measured may be categorized by the methods such as: maturity; multidimensional scorecards; value; benchmarking; modeling; and statistical analysis [31]. In terms of management/organizational perspective, there are several security metric categorizations. In [32], the Center for Internet Security (CIS) divided security metrics into three groups which are Management, Operations, or both. Chew et al. [33] grouped security metrics by Implementation, Effectiveness and Efficiency, and Business Impact. Savola [34] differentiated metrics into Management, Operational, and Technical. These categorizations may overlap as well as interrelate. However, these taxonomies tend to simplify complex socio-technical or practice-theory relationships [35].

Security metrics requirement. In a metrics system, several requirements of a good security metric are considered carefully and have been proposed by organizations and researchers. Jaquith [30] asserts that security metrics requirements should include consistently measured, cheap to gather, expressed as a cardinal number or percentage and using at least one unit of measure, and contextually specific. According to Wesner [36], security metrics should be SMART (Specific, Measurable, Actionable, Relevant, and Timely). Brotby [37] proposes PRAGMATIC (Predictive, Relevant, Actionable, Genuine, Meaningful, Accurate, Timely, Independent, Cheap). Herrmann [38] considers that a good security metric is one that possesses Accurate, Precise, Valid, and Correct characteristics.

Security metrics program. Once the security metrics have been decided by an organization for its system, a security metrics program has to be established to provide the organization with a map to manage, control, or improve the system security domains [39]. Several methods to build up a security metrics program are deployed. First, Payne [40] proposed Seven Steps model to establish security metrics including: defining the metrics program goal(s) and objectives; deciding metrics to generate; developing strategies for generating the metrics; establishing benchmarks and targets; determining metrics are reported; creating an action plan and act on it; and establishing a formal program review/refinement cycle. NIST also considered the metrics development and selection cycle via seven steps from identify stakeholders and interest to business mission impact [41].

Chew et al. [33] proposed five key components of making a metrics program plan: program initiation; development of information security metrics; analysis of information security metrics; reporting information security metrics; maintaining an information security metrics program. Campbell and Blades [42] listed five steps in a security metrics program: identifying the business drivers and objectives for the security metrics program; determining who your metrics are intended to inform and influence; identifying the types and locations of data essential for actionable security metrics; establishing relevant metrics; and establishing internal controls to ensure integrity of data and data assessments and to protect confidentiality.

3. Cloud security capability maturity model (CSCMM). To solve all above problems from cloud models and cyber security maturity models, we developed a Cloud Security Capability Maturity Mode (CSCMM) with two dimensions including domain and maturity level (Figure 3.1). The first dimension presents twelve cloud security domains. Each domain is a set of cyber security practices. The practices within each domain are the achievement objectives specific for cloud security domain. The second dimension shows four maturity levels which apply seperately to each domain. The maturity levels indicate a progression of maturity.

The model is built from a combination of existing cyber security standards, frameworks, and innovation. It provides the guidance to support the organizations implement and enhance their cyber security capabilities on cloud system. The model can be tailored for appropriate goals of different cloud service model (IPSaaS) and deployments (Public, Private, and Hybrid Cloud).

3.1. CSCMM domains. There is not a complete cloud security standard because cloud technology is evolving much faster than standards [43]. Therefore, creating a set of security domains just based on the current security standards is not adequate to take into account emerging issues and attack surfaces. For CSCMM, we choose a systematic review approach on existing cloud security models and standards, traditional security maturity models as well as trends in emerging technologies. Systematic review methodology is a means of evaluating and interpreting available research relevant to a particular research question, topic area, or phenomenon of interest [44]. As a result, we investigated fourteen security models including five traditional and nine cloud security models. We found twelve in twenty one security domains with the highest number of appearances in fourteen models (Figure 3.2). In which, eight security domains are from traditional maturity models and

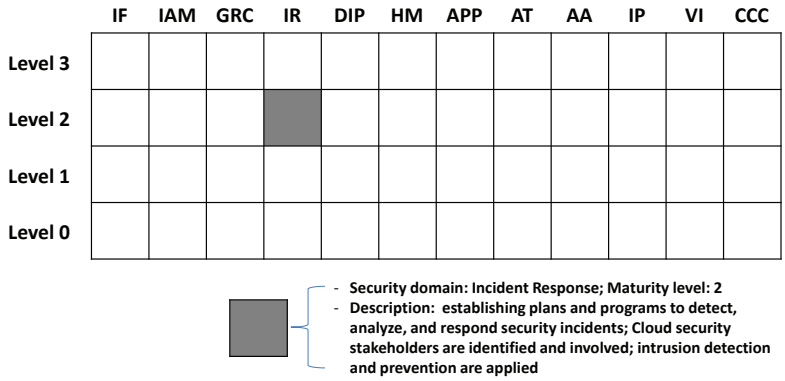


FIG. 3.1. *CSCMM Model Architecture*

standards including infrastructure and facilities security; identity and access management; governance, risk, and compliance; incident response and threat management; data and information protection; human resources management; security awareness and training; audit and accountability. There are four cloud specific security domains such as cloud connections and communication; operability and portability; virtualization; and application secuirty. Based on different perspective of security domains categories from ISO (strategic, tactical, and operational), CSA (governance, operational), IBM (Process, Technical, and Operational), and Karola (Technical, Social), we settle for these twelve security domains as they cover comprehensive aspects of cyber security and accommodate emerging security issues.

The main contents of these 12 domains are summarized below:

1. Infrastructure and facilities security (IF): The security of an IT system also depends on the security of its physical infrastructure and facilities. In the case of cloud computing, this extends to the infrastructure and facilities of the cloud service provider. The customer must get assurance from the provider that appropriate security controls are in place. ISO 27007 can be used to ensure protection against external and environmental threats like fire, floods, earthquakes, civil unrest or other potential threats that could disrupt cloud services; control of personnel working in secure areas; equipment security controls; and supporting utilities such as electricity supply, gas supply, telecommunications.

2. Identities and Access Management (IAM): This domain ensures authentication, authorization, and administration of identities. The main concerns of this domain are related to identity verification, granting a correct level of access to cloud resources, policy managements, and role-based access controls. The purpose of IAM is to prevent unauthorized access to physical and virtual resources as this can threaten the confidentiality, availability, integrity, and other properties of users services and data. These domains can be applied by standards or technologies such as LDAP (Lightweight directory Access Protocol) to provide access to directory servers and SAML 2.0 (Security Authorization Mark-up Language) for exchange of authentication and authorization data between security domains.

3. Governance, Risk, and Compliance management (GRC): This domain focuses on establishing, operating, and maintaining cyber security risk management programs that identify, analyse, and mitigate cyber security risk to the organization. This means governance and compliance policies and procedures are established to protect stakeholders property. This covers implementations of compliance following regulatory requirements between stakeholders. Compliance management is to maintain and provide compliance. It relates to execution of internal security policies, and different compliance requirements such as regulatory, legislative.

4. Incident response (IR): This domain concentrates on incident detection, response, notification, and remediation. The major concerns in incident response are related to establishing and maintaining plans, procedures, and technologies to detect, analyse, and respond to cyber security incidents and events. The incident response lifecycle as expressed in the National Institute of Standards and Technology Computer Security Incident Handling Guide (NIST 800-61) should be used in this domain.

5. Data and Information protection (DIP): Data protection is one of the critical security challenges in cloud

computing. Control of data and compensating controls can be used to tackle the loss of physical control when moving data to the cloud. The concern of information management is who has onus for data confidentiality, integrity, and availability. Therefore, security controls as expressed in ISO 27002 including asset management, access control and cryptography can be applied. Other technologies such as HTTPS for regular connections from cloud services over the internet, VPN using IPSec or SSL for connections also can be used for implementing this domain. Moreover, encryption keys should be used by KMIP (the Key Management Interoperability Protocol) that supports a standardized way to manage encryption keys.

6. Human resource management (HM): People are often described as the weakest entity in any security system. This domain focuses on human resource process, from pre-employment, during employment, and through termination, to ensure that policies and procedures are in place to address security issues. The three areas of human resources security concerned are prior to employment; during employment; termination and change of employment. Human Resources Security in ISO 27002:2013 (Information Security Management) can be used for this domain.

7. Cloud application security (APP): This domain focuses on determining the application software on which type of cloud platform (SaaS, PaaS, or IaaS) for securing. The Open Web Application Security Project (OWASP) or Secure Software Development Life Cycle (SSDLC) can support cloud service entities to secure application running on cloud systems. In terms of technologies and techniques in cloud application security, firewall can be used to control access. VPNs can be considered to limit access to application to users for these domains.

8. Security awareness and training (AT): This domain aims to create a culture of security and ensure the ongoing suitability and competence of all personnel. Consistent training throughout the entire process ensures that employees and contractors are fully aware of their roles and responsibilities and understand the criticality of their actions in protecting and securing both information and facilities.

9. Audit and Accountability (AA): This domain aims to provide information about roles, responsibilities, and compliance regarding auditing. It addresses auditing of security controls including checking for proper server maintenance and controls to make sure that it is properly done and security policies are being enforced. The policy may set the level and detail of auditing and specify types of events to be audited. The major procedures of this domain are auditable events; content of audit records, audit processing and monitoring; audit reduction and report generation; protection of audit information; and audit retention.

10. Interoperability and portability (IP): This domain is one of the special domains in cloud computing. It is the ability to move data/services from one provider to another, or bring it entirely back in-house. To ensure this domain, we can use open virtualization formats to provide interoperability, while virtualization can help to remove concerns about physical hardware, distinct differences exist between common hypervisors. It deals with different technologies virtual machine images are captured and ported to new cloud providers such as Distributed Management Task Force (DMTF) and Open Virtualization format (OVF).

11. Virtualization and isolation (VI): This domain focuses on the security issues related to system/hardware virtualization, rather than a more general survey of all forms of virtualization. This domain is associated with multi-tenancy, VM isolation, VM co-resident, hypervisor vulnerabilities, and other virtualized artefacts. Isolation is the technique used to protect each entity within the cloud infrastructure component of a system from unwanted interferences. Isolation is used to identify virtual and physical boundaries, partition containers, processes or logical functional entities, and isolate policy-based security violations.

12. Cloud connection and communication security (CCC): A cloud service provider must allow legitimate network traffic and block malicious network traffic. However, unlike many other organizations, a cloud service provider may not necessarily know what network traffic its customers plan to send and receive. Nevertheless, customers should expect certain external network perimeter safety measures from their cloud providers. For this domain, ISO/IEC 2703332 standards can be used to provide detailed guidance on implementing the network security controls that are introduced in ISO/IEC 27002.

In these twelve domains, we integrate isolation aspects into virtualization domain to generate new domain namely virtualization and isolation and introduce domain interoperability portability as a new domain. It is clear that virtualization and isolation have been important techniques in cloud security. Virtualization is considered as the cloud enabling technology and hence it is at the centre of cloud security. However, with emerging attacks

ID	Domains/Models	CSA	CSCC	ENISA	IBM	CISCO	ISIMC	FedRAMP	PCIDSS	SANS	SSE-CMM	ES-CMM	RMM	ISO	NIST-CSF	Number
1	Infrastructure and facilities security (IF)	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	13
2	Identity and access management (IAM)	✓	✓	✓	✓		✓	✓		✓	✓	✓		✓	✓	11
3	Governance, Risk, and Compliance (GRC)	✓	✓	✓	✓		✓	✓	✓		✓	✓	✓		✓	11
4	Incident response (IR)	✓		✓			✓	✓	✓	✓		✓	✓		✓	9
5	Data and information protection (DIP)	✓	✓	✓	✓	✓			✓	✓					✓	8
6	Human resources management (HM)		✓	✓	✓	✓		✓				✓		✓		7
7	Application security (APP)	✓	✓	✓	✓	✓	✓				✓					7
8	Security awareness and training (AT)			✓			✓	✓			✓		✓		✓	6
9	Audit and accountability (AA)	✓					✓	✓			✓	✓				5
10	Interoperability and portability (IP)	✓		✓			✓									3
11	Virtualization and isolation (VI)	✓				✓	✓									3
12	Cloud connection and communication (CCC)		✓	✓	✓											3

FIG. 3.2. *The appearance of security domains in security model*

recently on the virtualization layer, this domain has to be taken seriously. Isolation techniques have emerged as a new approach for securing cloud computing. The development of isolation theory with assessing process is necessary.

3.2. Security maturity levels. To investigate the common features of each maturity level in previous security maturity models, we compared ten prominent professional security maturity models (Figure 3.3). As a result of this investigation, we adopt four maturity levels (SMLs) for our CSCMM model. Maturity levels are identified by the following attributes: (1) the SMLs apply independently to each domain. For instance, an organization could be implementing at SML1 in one domain, SML2 in another domain; (2) the maturity level of a domain is determined by the minimum of all security practices implemented in that domain. For example, to gain security maturity level at SML2 in one domain, the organization has to implement all the security practices in SML1 and SML2; (3) SML achievement should align with business objectives and organizations security strategy.

Expressed below are common features that define each maturity level.

- SML0 (Undefined): at this level, organizations are at the starting point with a commitment to establish a security maturity assessment model. They have no plans to check or test security processes.

- SML1 (Initiated): at this level, most organizations focus on basic security practices. Some basic security physical hardware devices or networks need to be implemented on IaaS, basic protection on virtual machine monitor, access control and encryption on PaaS, basic application security and multi-tenancy on SaaS.

- SML2 (Managed): at this level, organizations focus on building and planning Information Security programs and apply cloud security standards. Cloud security stakeholders such as provider, consumer, and third-party are identified and involved. Cloud security activities need to be guided by policies. Some cloud automatic security tools are applied such as intrusion detection and prevention systems. Especially, security metrics system needs to be applied at this level to support security decision making. For IaaS, security mechanisms to protect network and data are applied to achieve selected security standards compliance. For PaaS, it is ensured that the virtual machine monitor needs to be protected by higher security policies. For SaaS, automatic security system for web-based, software, or database need to be implemented.

- SML3 (Optimized): it is defined as the highest maturity level. This is real-time protection level. All the security program support 24/7 staffed operations and fully automated. It is assured that all security policies and procedures are implemented. This is the ideal cloud security status with optimal use of resources from facilities, time to costs and human. This level is called resilience when the organization can detect and tackle with security threats automatically proactively and the time to achieve resilience status is almost zero. All people in the organization have adequate skills and knowledge about security on cloud.

	Cyber Security Maturity Models	Organizations or Author	Purposes and Strengths	Maturity Levels				
				1	2	3	4	5
1	**Information security management system (ISMS-ISO 27001), 2005**	ISO	Information security risk management through security standards	Performed	Managed	Established	Predictable	Optimized
2	**Information Security Management Maturity Model (ISM3), 2007**	ISM3 Consortium	Prevent and mitigate incidents and Optimise the use of information, money, people, time and infrastructure	Undefined	Defined	Managed	Controlled	Optimized
3	**Information Security Maturity Model (ISM2), 2007**	NIST-PRISMA	Provides a framework for review and measure the information security posture of an information security program	Polices	Procedures	Implemented	Tested	Integrated
4	**Gartner's Information Security Awareness Maturity Model (GISAMM), 2009**	Gartner	Security awareness, and risk management in large international organizations	Blissful ignorance	Awareness	Corrective	Operations excellence	
5	**Information Security Framework (ISF), 2009**	IBM	Security gap analysis between business and technology	Initial	Basic	Capable	Efficiency	Optimizing
6	**Resilience Management Model (RMM), 2010**	CERT	A capability-focused process model for managing operational resilience	Incomplete	Performed	Managed	Defined	
7	**Community Cyber Security Maturity Model (CCSMM), 2011**	White	Community effort and communication capability in communities	Initial	Advanced	Self-Assessed	Integrated	Vanguard
8	**NICE's Cyber Security Capability Maturity Model, 2012**	The US DHS	Workforce planning for cyber security best practices	Limited	Progressing	Optimized		
9	**Cyber Security Framework (CSF-NIST), 2014**	NIST	Improves federal critical infrastructure through a set of activities designed to develop individual profiles for operators	Identify	Protect	Detect	Respond	Recover
10	**Cyber Security Capability Maturity Model (C2M2), 2015**	Curtis	Assessment of implementation and management in Critical Infrastructure	Not performed	Initiated	Performed	Managed	

FIG. 3.3. *Discovering Cyber Security Maturity Models*

4. Security metrics framework. To assess the maturity level of CSCMM model in general and a security domain in particular, we propose a security metrics framework with the following steps (Figure 4.1).

Inputs. This first step describes the requirements for the security metrics framework: security practices and activities, goals and objectives, security requirements. A set of security practices for a particular domain or multiple domains is defined and/or selected. This depends on the demand of upper management or the schedule of assessment process of the CSCMM model. These securities then determine what to measure. What-to-measure may be one security activity or several security activities from the selected domains. Stakeholders are identified which include upper managers who decide on information requirements, managers who carry out the directive, practitioners who implement the security metrics, and security metrics consumers. Goals and Objectives define the goals and objectives of security metrics plan or program from the stakeholders viewpoint.

Metric plan. Classification of security activities or practices is also necessary to indicate the type of measurement (governance, management, operational, and technical) and to decide on the metrics plan and the method to measure as security metrics should be SMART [36] or PRAGMATIC [37]. Security metrics components identification identifies the elements or dimensions related to the metrics. These may include real-virtual, infrastructures, and interaction of entities in the (cloud) cyber space, and other factors such as cost, time, threats, and vulnerabilities. Determination of measuring methods is based on the qualitative or quantitative nature of the security practices. Quantitative metrics are usually based on mathematical models and numerical data. The unit of measurement for each component of security metrics program is then derived. Data collection has to be planned to meet the characteristic requirements such as obtainable, cheap to collect, quantitative express, automatically.

Measuring. Relevant and measurable metrics have already determined and selected from previous steps, this step carries out the actual measurement according to the measuring method and the data collection plan. In general, a security metric is a function of its measured components:

$$x = f(x_1, x_2, x_3, \ldots, x_n) \tag{4.1}$$

in which, $x_1, x_2, x_3, ..., x_n$ are security metric components and x could be a countable value based on a maturity benchmarking (next step). f is a function of the specification of security metrics identified in the metric plan. If x does not yield a value or it is impossible to implement the measurement one has to go back to the metric plan step redefine the set security components and their impacts.

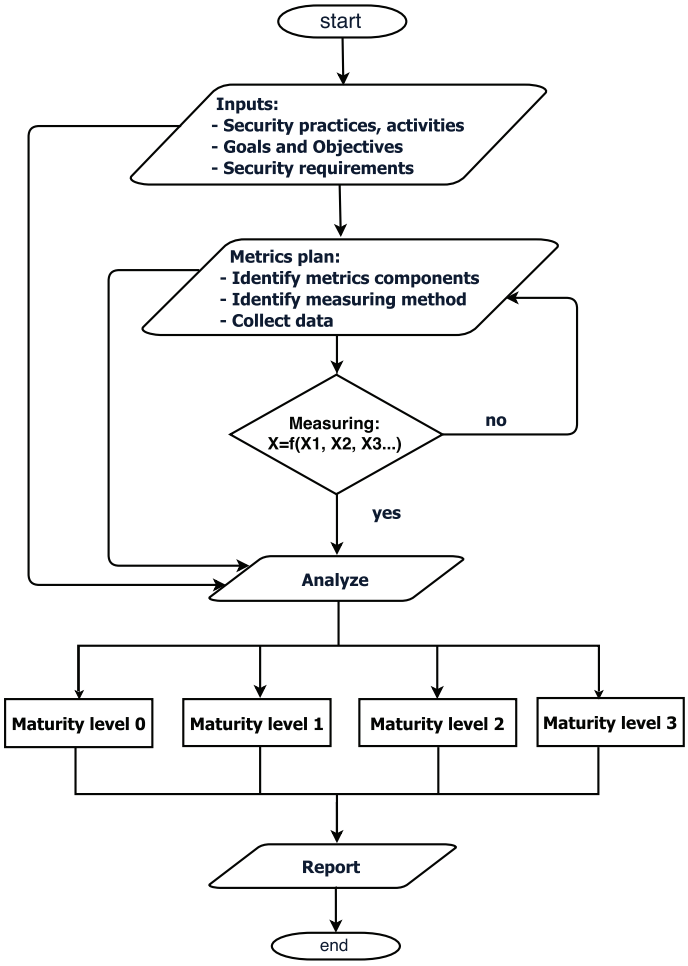


FIG. 4.1. *CSCMM metrics framework diagram*

Analyze. This step consists of several operations such as holistic analysis, interpretation, and consolidation. Holistic analysis means that the analysis takes into account not only the measured metrics but also the elements of the inputs and the metric plan steps of the metrics framework. This is important as some quantitative metrics lose their original meanings when reduced to a pure numerical number. Interpretation of the obtained metrics is to decipher the true security status of the cyber space under protection. Interpretation also provides the reasons and their impact on the measured result. The effectiveness and efficiency of the proposed metrics should be evaluated.

Maturity level determination. Benchmarking is the process of comparing ones own performance and practices against peers within the industry or noted best practice organizations outside the industry. Benchmarks can be used, for example, to determine a minimum essential configuration for workstations, servers, laptops, routers, firewalls, and other network devices or for the holistic system. The method for assigning maturity level depends on the specification of the security metrics. It could be assigned as a percentage range from Level 0 (say, 0-25%) to Level 4 (say, 75-100%); a weighted value; a value interval, or times to security incident response from months (level 0), days (level 1), hour (level 2), to real-time (level 3) [45].

Report. The last step is reporting that shows and informs the ultimate impact and consequences to metrics consumers. All steps of the metrics need to be described. The frequency of reports depends on requirement of the organization and its upper managers. On the one hand, the report provides the assessed security status of the cloud system relates and explain clearly the impact of the security status to the management on the organization

business plan and direction. On the other hand, to the security experts and practitioners, the report identifies security weaknesses and suggests action plans for remedy and provides a roadmap for strengthening the security of the system.

5. The selection of advanced security quantitative metrics for CSCMM. With the proposed security metrics framework, the overall security assessment can be balanced and complemented between existing qualitative assessment for senior managers of an organization and quantitative assessment for its security experts. In terms of the qualitative assessment, capability maturity model theory provides senior managers with a sound picture of security compliance of their system in terms of practices but it does not relate well the impact of the security assessment to their business plan and direction. In terms of quantitative assessment, advanced security metrics allow mappings between the outcome of security assessment and costs/benefits to the organization. Furthermore, good quantitative security metrics allow the identification of a specific domain or an individual practice of the model and suggest appropriate security measures for achieving a higher level of maturity.

Among many quantitative security metrics, Mean Failure Cost (MFC) metrics [46] is an excellent candidate metric for CSCMM. MFC is the predictive quantitative metric that quantifies the costs each (among many) stakeholder needs to invest to the mission for better security or the benefits the stakeholder stands to lose due to the lack of security. MFC is considered as an advanced security metrics for a number of reasons. First, it includes the stakeholders, the impact of security properties on stakeholders, and the threats that can affect system. Second, it can embrace traditional metrics such Mean time to failure (MTTF), Mean Time To Explore (MTTE), and Mean Time Between Breaches (MTBB). Third, it meets many essential security metrics requirements such as SMART or PRAGMATIC.

In addition, the assessment process in the CSCMM model can deploy other state-of-art quantitative metrics including check-list based, state-based stochastic, microaggregation technique, fuzzy analytic hierarchy, attack graph based, Dynamic Bayesian Network (DBN) based, formal methods, and tree weighting. Check-list-based metrics propose an advanced security measurement system that reflects the characteristics of each field (critical infrastructure facilities) to achieve effective information security management [47]. State-based stochastic metrics focus on progression of an attack process over time. This applies for 4 types of significant attacks: Buffer Overflow, Man-in-the-middle, SQL injection, and Traffic Sniffing [48]. Microaggregation is the technique to protect cloud data through anonymity in order to prevent exposure of person's identity [49]. Fuzzy analytic hierarchy presents a quantitative framework based on Fuzzy Analytic Hierarchy Process (FAHP) to quantify the security performance of an information system [50]. Attack graphs based provides a method for quantitatively analysing the security of a network using attack graphs that are populated with known vulnerabilities and likelihoods of exploration and then exercised to obtain a metric of the overall security and risk of the network [51]. Dynamic Bayesian Network (DBN) based model is used to capture the dynamic nature of vulnerabilities that change overtime. An attack graph is converted to a DBN by applying conditional probabilities to the nodes, calculated from the Common Vulnerabilities Scoring System [52]. Formal methods are being used for verification of cloud computing systems including verification of security in partitioned cloud, firewall, and big data [53]. Tree weighting proposes an initial framework for estimating the security strength of a system by decomposing the system into its security sensitive components and assigning security scores to each component [54].

One of the quantitative metrics proposed for CSCMM is called the Mean Remediate Time (MRT). The metric quantifies the costs (here, in term of time) each cloud stakeholder has to spend to remediate as a result of a security breach or failure. The metric relates the stakeholders cost vector to the probability of a realized threat vector through three multiplicative matrices: stakeholder, threat class, and threats matrix. To arrive at the quantitative MRT, a new cloud security stakeholder model is introduced to identify cloud security stakeholders and their interrelationships. In addition, a security threat probability distribution proposed based on attack-graph, Common Vulnerability Scoring System (CVSS) and Markov chain theory. Clearly, quantitative metrics are not applicable to all cloud domains as for some domains existing qualitative metrics are more appropriate for security compliance. However, the essence of a quantitative metric is that it allows the security assessors to ascertain the security level of each domain and of the overall cloud so that senior management of the organization can make appropriate decisions in terms of sound security investment, meaningful system upgrade accordance to their business plan. The quantitative metric must also allow security practitioners or security managers to

identify the security weaknesses of the system and provide the roadmap for security implementation.

6. Conclusion. This paper reviewed and revised a number of security concepts and models including cloud security models, security capability maturity models, and security metrics. The security capability maturity models are of particular interest as they systematically cover all important aspects of a cyber-infrastructure. The paper proposed a Cloud Security Capability Maturity Model that includes cloud-specific security domains and provides quantitative assessment of the overall security of the cloud under consideration. To support the measuring of security maturity level, a security metrics framework was introduced. This framework includes relevant quantitative metrics for measurable assessment. It presents a balance assessment of the overall security of an organisation/system qualitatively and quantitatively. For senior managers, CSCMM offers a meaningful security assessment of the security status of their infrastructure for making decision concerning business plan and direction. For security experts or practitioners, CSCMM with its quantitative metrics enables proactive measures and responsive actions. The paper also suggested future research with advanced metrics that involve various stakeholders, components of cloud security systems.

REFERENCES

[1] Humphrey, *Cmm*, IEEE, 1 (1989).
[2] P. D. Curtis and N. Mehravari, *Evaluating and improving cybersecurity capabilities of the energy critical infrastructure*, in Technologies for Homeland Security (HST), 2015 IEEE International Symposium on, pp. 1–6.
[3] N. Wiener, *Cybernetics or Control and Communication in the Animal and the Machine*, vol. 25, MIT press, 1961.
[4] S. Cirani, M. Picone, and L. Veltri, *A session initiation protocol for the internet of things*, Scalable Computing: Practice and Experience, 14 (2014), pp. 249–263.
[5] M. Whitman and H. Mattord, *Management of information security*, Cengage Learning, 2013.
[6] R. Von Solms and J. Van Niekerk, *From information security to cyber security*, computers & security, 38 (2013), pp. 97–102.
[7] M. Gasser, *Building a secure computer system*, Van Nostrand Reinhold Company New York, NY, 1988.
[8] D. Craigen, N. Diakun-Thibault, and R. Purse, *Defining cybersecurity*, Technology Innovation Management Review, 4 (2014).
[9] M. C. Lacity, *Advanced outsourcing practice: Rethinking ito, bpo and cloud services*, Palgrave Macmillan, 2012.
[10] A. Behl and K. Behl, *An analysis of cloud computing security issues*, in Information and Communication Technologies (WICT), 2012 World Congress on, pp. 109–114.
[11] D. Catteddu, *Cloud Computing: benefits, risks and recommendations for information security*, Springer, 2010, pp. 17–17.
[12] C. S. C. Coucil, *Security for cloud computing ten steps to ensure success version 2.0*, report, March, 2015.
[13] W. R. Claycomb and A. Nicoll, *Insider threats to cloud computing: Directions for new research challenges*, in 2012 IEEE 36th Annual Computer Software and Applications Conference, pp. 387–394.
[14] S. Farhan Bashir and S. Haider, *Security threats in cloud computing*, in Internet Technology and Secured Transactions (ICITST), 2011 International Conference for, pp. 214–219.
[15] C. S. ALLIANCE, *The treacherous twelve - cloud computing top threats in 2016*, 2016.
[16] ENISA, *Security standards for cloud usage*, report, August 2014.
[17] J. Archer and A. Boehm, *Security guidance for critical areas of focus in cloud computing*, Cloud Security Alliance, 2 (2009), pp. 1–76.
[18] G. Brunette and R. Mogull, *Security guidance for critical areas of focus in cloud computing v2. 1*, Cloud Security Alliance, (2009), pp. 1–76.
[19] C. Alliance, *Security guidance for critical areas of focus in cloud computing v3. 0*, Cloud Security Alliance, (2011).
[20] B. Swain, P. Agcaoili, M. Pohlman, and K. Boyle, *Cloud controls matrix*, 2010.
[21] J. Allen and N. Mehravari, *How to be a better consumer of security maturity models*, report, Citeseer, 2014.
[22] N. T. Le and D. B. Hoang, *Can maturity models support cyber security?*, in 2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC), pp. 1–7.
[23] M. Siponen and R. Willison, *Information security management standards: Problems and solutions*, Information & Management, 46 (2009), pp. 267–270.
[24] B. Stevanovi, *Maturity models in information security*, International Journal of Information, 1 (2011).
[25] P. E. Black, K. Scarfone, and M. Souppaya, *Cyber security metrics and measures*, Wiley Handbook of Science and Technology for Homeland Security, (2008).
[26] B. Bates, K. M. Goertzel, and T. Winograd, *Measuring Cyber Security and Information Assurance: A State-of-the Art Report*, Information Assurance Technology Analysis Center, 2009.
[27] W. Thomson, *Lord kelvin: Electrical units of measurement. popular lectures and addresses*, 1889.
[28] W. S. Humphrey, *A discipline for software engineering*, Addison-Wesley Longman Publishing Co., Inc., 1995.
[29] R. Trapero, L. Jesus, and S. Neeraj, *Quantifiably Trusting the Cloud: Putting Metrics to Work*, IEEE Security & Privacy, 3 (2016), pp. 73–77.
[30] A. Jaquith, *Security metrics*, Pearson Education, 2007.
[31] W. K. Brotby, *Information security management metrics*, A definitive guide to effective security monitoring and, (2009).

[32] C. f. I. Security, *The cis security metrics*, 2010.
[33] E. Chew, M. Swanson, K. Stine, N. Bartol, A. Brown, and W. Robinson, *Performance measurement guide for information security*, 2008.
[34] R. Savola, *Towards a security metrics taxonomy for the information and communication technology industry*, in Software Engineering Advances, 2007. ICSEA 2007. International Conference on, IEEE, pp. 60–60.
[35] S. Kowalski, R. Barabanov, and L. Yngstrm, *Information security metrics: Research directions*, (2011).
[36] J. P. Ravenel, *Effective operational security metrics*, Information Systems Security, 15 (2006), pp. 10–17.
[37] W. K. Brotby and G. Hinson, *Pragmatic security metrics: applying metametrics to information security*, CRC Press, 2013.
[38] D. S. Herrmann, *Complete guide to security and privacy metrics: measuring regulatory compliance, operational resilience, and ROI*, CRC Press, 2007.
[39] S. E. Schimkowitsch, *Key Components of an Information Security Metrics Program Plan*, thesis, 2009.
[40] S. C. Payne, *A guide to security metrics*, SANS Security Essentials GSEC Practical Assignment Version, 1 (2010).
[41] W. Jansen, *Directions in security metrics research*, Diane Publishing, 2010.
[42] G. Campbell and M. Blades, *Building a metrics program that matters*, Journal of healthcare protection management: publication of the International Association for Hospital Security, 30 (2014), p. 116.
[43] B. Duncan and M. Whittington, *Compliance with standards, assurance and audit: does this equal security?*, in Proceedings of the 7th International Conference on Security of Information and Networks, ACM, p. 77.
[44] B. Kitchenham, *Procedures for performing systematic reviews*, Keele, UK, Keele University, 33 (2004), pp. 1–26.
[45] R. Lentz, *Security intelligence maturity model*, 2015.
[46] M. Jouini, A. B. Aissa, L. B. A. Rabai and A. Mili, *Towards quantitative measures of Information Security: A Cloud Computing case study*, International Journal of Cyber-Security and Digital Forensics (IJCSDF), 1 (2012), pp. 248-262
[47] Y. You, I. Cho, and K. Lee, *An advanced approach to security measurement system*, The Journal of Supercomputing, 72 (2016), pp. 3443–3454.
[48] J. Almasizadeh and M. A. Azgomi, *A stochastic model of attack process for the evaluation of security metrics*, Computer Networks, 57 (2013), pp. 2159–2180.
[49] S. Tonni, M. Rahman, S. Parvin, and A. Gawanmeh, *Securing Big Data Efficiently through Microaggregation Technique*, in Distributed Computing Systems Workshops (ICDCSW), 2017 IEEE 37th International Conference, pp. 125–130.
[50] S. Thalia, A. Tuteja, and M. Dutta, *Towards quantification of information system security*, Springer, 2011, pp. 225–231.
[51] S. Noel, S. Jajodia, L. Wang, and A. Singhal, *Measuring security risk of networks using attack graphs*, International Journal of Next-Generation Computing, 1 (2010), pp. 135–147.
[52] M. Frigault, L. Wang, A. Singhal, and S. Jajodia, *Measuring network security using dynamic bayesian network*, in Proceedings of the 4th ACM workshop on Quality of protection, ACM, pp. 23–30.
[53] A. Gawanmeh, and A. Ahmad, *Challenges in formal methods for testing and verification of cloud computing systems*, Scalable Computing: Practice and Experience, 3 (2015), pp. 321–332.
[54] C. Wang and W. A. Wulf, *Towards a framework for security measurement*, in 20th National Information Systems Security Conference, Baltimore, MD, pp. 522–533.

Edited by: Amjad Gawanmeh
Received: May 2, 2017
Accepted: Oct 28, 2017

DOI 10.12694/scpe.v18i4.1330
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 18, Number 4, pp. 291–311. http://www.scpe.org

AUTOMATIC AND SCALABLE DATA REPLICATION MANAGER IN DISTRIBUTED COMPUTATION AND STORAGE INFRASTRUCTURE OF CYBER-PHYSICAL SYSTEMS

ZHENGYU YANG *, JANKI BHIMANI †, JIAYIN WANG ‡, DAVID EVANS §, AND NINGFANG MI ¶

Abstract. Cyber-Physical System (CPS) is a rising technology that utilizes computation and storage resources for sensing, processing, analysis, predicting, understanding of field-data, and then uses communication resources for interaction, intervene, and interface management, and finally provides control for systems so that they can inter-operate, evolve, and run in a stable evidence-based environment. There are two major demands when building the storage infrastructure for a CPS cluster to support above-mentioned functionalities: (1) high I/O and network throughput requirements during runtime, and (2) low latency demand for disaster recovery. To address challenges brought by these demands, in this paper, we propose a complete solution called "AutoReplica" – an automatic and scalable data replication manager in distributed computation and storage infrastructure of cyber-physical systems, using tiering storage with SSD (solid state disk) and HDD (hard disk drive). Specifically, AutoReplica uses SSD to absorb hot data and to maximize I/Os, and its intelligent replication scheme further helps to recovery from disaster. To effectively balance the trade-off between I/O performance and fault tolerance, AutoReplica utilizes the SSDs of remote CPS server nodes (which are connected by high speed fibers) to replicate hot datasets cached in the SSD tier of the local CPS server node. AutoReplica has three approaches to build the replica cluster in order to support multiple SLAs. AutoReplica automatically balances loads among nodes, and can conduct seamlessly online migration operation (i.e., migrate-on-write scheme), instead of pausing the subsystem and copying the entire dataset from one node to the other. Lastly, AutoReplica supports parallel prefetching from both primary node and replica node(s) with a new dynamic optimizing streaming technique to improve I/O performance. We implemented AutoReplica on a real CPS infrastructure, and experimental results show that AutoReplica can significantly reduce the total recovery time with slight overhead compared to the no replication cluster and traditional replication clusters.

Key words: Cyber Physical Systems Infrastructure, Replication, Backup, Fault Tolerance, Device Failure Recovery, Distributed Storage System, Parallel I/O, SLA, Cache and Replacement Policy, Cluster Migration, VM Crash, Consistency, Atomicity

AMS subject classifications. 68M14, 68N30, 68P20

1. Introduction.

With the rise of Cloud Computing and Internet of Things, as an enabling technology, Cyber-Physical Systems (CPS) is increasingly reaching almost everywhere nowadays [1, 2, 3, 4]. CPS uses computing, communication, and control methods to operate intelligent and autonomous systems that can provide using cutting edge technologies. That is to say, CPS is the integration of computation, networking, and physical processes. As illustrated in Fig. 1.1, a typical CPS structure has the following three stages:

- **Data Capture Stage:** consists of relative lightweight "field devices" (also called "CPS clients") such as embedded computers, sensors, network and other mobile CPS devices.
- **Data Management Stage:** is responsible for multi-stream data collection, data storage, preliminary process, sharing control. AutoReplica is working on this stage.
- **Data Process Stage:** analyzes the streaming data, makes decisions, and sends feedbacks to CPS clients.

In modern CPS use cases, huge amount of data are needed to be stored and processed on the CPS cluster, and the corresponding I/O pressure will be mainly put on the "Data Management Stage". In real implementation of a CPS cluster, there are two challenges in this stage: The first challenge is related to the I/O speed requirement in large-scale CPS. Traditional HDDs are not efficient for high speed I/O requires [5, 6]. Therefore, high speed SSDs are often utilized in CPS storage system, as shown in Fig. 1.2 left subfigure, where a CPS server node can have multiple storage devices such as SSDs, performance-oriented HDDs and archive-oriented HDDs as shown

* Dept. of Electrical & Computer Engineering, Northeastern University, 360 Huntington Ave., Boston, MA, USA, 02115 (yangzy1988@coe.neu.edu).
† Dept. of Electrical & Computer Engineering, Northeastern University, 360 Huntington Ave., Boston, MA, USA, 02115 (bhimani@ece.neu.edu).
‡ Dept. of Computer Science, University of Massachusetts Boston, 100 Morrissey Boulevard, Boston, MA, USA 02125 (jiayin.wang001@umb.edu)
§Samsung Semiconductor Inc., Memory Solution Research Lab, Storage Software Group, San Diego, CA,USA 92121 (david.evans@samsung.com)
¶ Dept. of Electrical & Computer Engineering, Northeastern University, 360 Huntington Ave., Boston, MA, USA, 02115 (ningfang@ece.neu.edu).

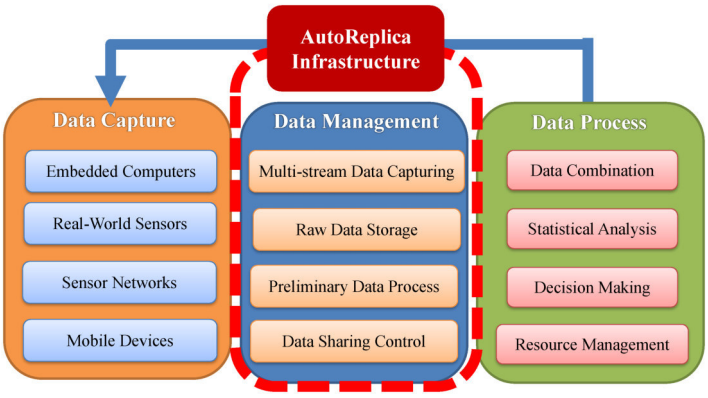


FIG. 1.1. *Three-stage data flow and components of CPS implementation.*

in the dash box. Above that, multiple virtual machines (VMs) running CPS platform applications are hosted by the hypervisor software, and all of them are sharing the storage pools. In Fig. 1.2, the right side figure further illustrates that in order to speed up the I/O performance of the storage system, SSDs are used to cache hot data, and HDDs are designed to host backend cold data.

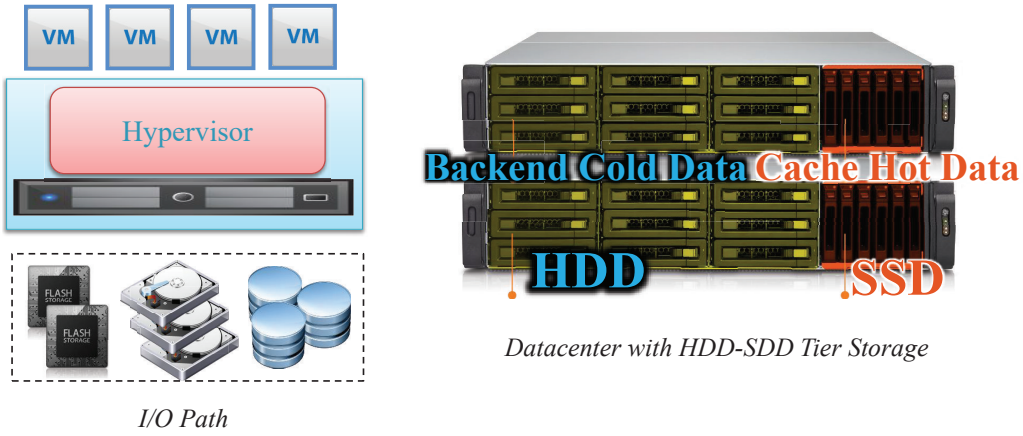


FIG. 1.2. *Storage architecture of each node.*

The second challenge is the problem of data recovery from different types of disasters. Data loss and delay caused by disasters will dramatically reduces the data availability and consistency, which are very critical for CPS applications. To address this challenge, replication technique – a process of synchronizing data across multiple storage nodes – is often used to provides redundancy and increases data availability from the loss of a single storage node [7, 8, 9, 10].

However, since redundancy brings overheads in terms of network traffic, I/O performance, storage space, and consistency maintenance, we need to balance the replication and performance [11, 12]. In piratical, SSDs are often used as the *write back* cache to improve to I/O speed, having only one up-to-date copy on SSD is not acceptable for high SLA (Service-Level Agreement) demand use cases such as bank, stock market, and military CPS networks. According to the study [13], compare to HDD, SSD is relatively not a "safe destination" though it can preserve the data after power off. Therefore, we focus on replicating only cached datasets in the SSDs, and now the main problem is *"where to store replicas of those datasets cached in the SSD while not downgrading the performance?"*

Motivated by this, we propose a complete solution called "AutoReplica", which is an automatic and scalable data replication manager designed for distributed CPS infrastructure with SSD-HDD tiering storage systems. AutoReplica maintains replicas of local SSD cache in the remote SSD(s) connected by high speed fibers, since the access speed of remote SSDs can be faster than that of local HDDs. AutoReplica can automatically build and rebuild the cross-node replica structure following three approaches designed based on different SLAs. AutoReplica can efficiently recover from different disaster scenarios (covers CPS service virtual machine crash, device failures and communication failures) with limited and controllable performance downgrades with a lazy migrate-on-write technique called "fusion cache", which can conduct seamlessly online migration to balance loads among nodes, instead of pausing the subsystem and copying the entire dataset from one node to the other. Finally, AutoReplica supports parallel prefetching from both primary node and replica node(s) with a novel dynamic optimizing streaming technique to further improve I/O performance. We implemented AutoReplica on VMware ESXi platform, and experimental results based on real CPS workloads show that AutoReplica can significantly improve the performance with slight or even less overheads compared to other solutions.

The remainder of this paper is organized as follows. Sect. 2 presents the topological structure of AutoReplica cluster. Sect. 3 introduces AutoReplica's cache and replacement policy, including the new "fusion cache" technique. Sect. 4 describes recovery policies under four different scenarios. Sect. 5 discusses the parallel prefetching scheme. Sect. 7 shows the experimental results. Finally, we summarize the paper in Sect. 8.

2. Topological Structure Of Datacenter. We first introduce topological structure of AutoReplica cluster [14, 15, 16]. As illustrated in Fig. 2.1, there are multiple nodes in the cluster, and each node is a physical host which runs multiple CPS service virtual machines (VMs). In our prototype, we use VMware's ESXi [17] to host VMs. Inside each node, there are two tiers of storage devices: SSD tier and HDD tier. The former tier is used as the cache and the latter tier is used as the backend storage. Each storage tier contains one or more SSDs or HDDs, respectively. RAID mode disks can also be adopted in each tier. SSD and HDD tiers in each node are shared by VMs and managed by the hypervisor.

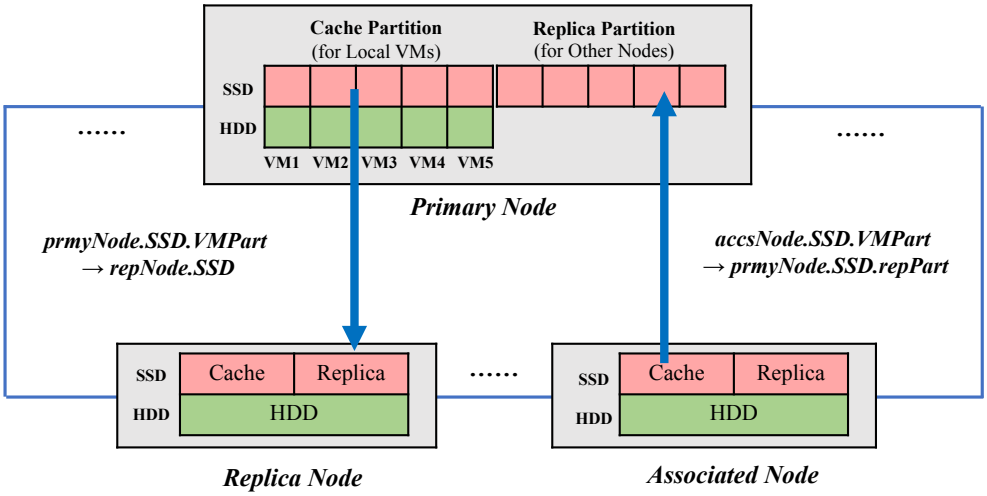


FIG. 2.1. *An example of the structure of AutoReplica's datacenter.*

Since nodes are connected by high speed fiber channels, the remote SSD access speed (including the network delay) can be even faster than local HDD access speed. Thus, to utilize remote SSDs as replica destination, inside the SSD tier, we set two partitions: "Cache Partition" (for the local VMs), and "Replica Partition" (for storing replica datasets from other nodes SSD cache). AutoReplica uses *write back cache policy* to maximize I/O performance, since writing through to HDD will slow down the I/O path. However, as mentioned, SSD is relatively vulnerable and not cannot be equally trusted as a "safe destination" like HDD, though SSD can preserve the data after power off. Therefore, AutoReplica maintains additional replicas in the remote SSDs

to prepare for recoveries for failures. In fact, we still can use local HDD as the second replica device for those extremely high SLA nodes, which will be discussed in Sect. 2.1. Based on these facts, we propose three approaches to setup the topological structure of the datacenter clusters, focusing on *"how to select replica nodes?"*, *"how many replicas nodes do we need?"*, and *"how to assign replicas?"*.

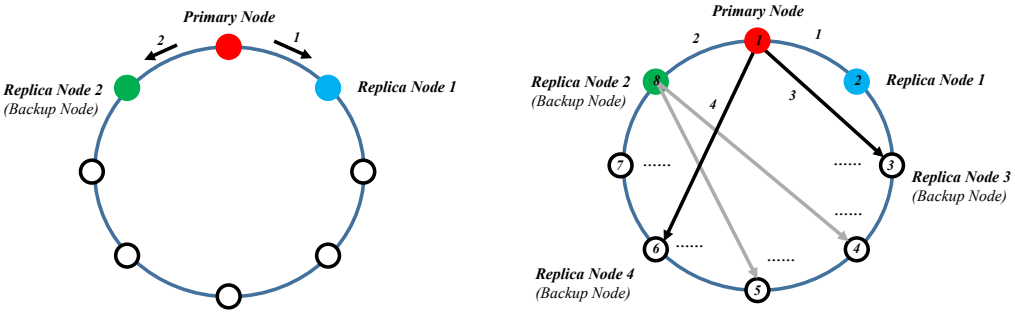


FIG. 2.2. *Examples of (a) Ring and (b) Network approaches.*

2.1. Ring Approach. Our first approach is a directed logical "Ring" structure, which can be either user-defined or system-defined. A system-defined ring is based on geographic distance parameters (e.g., I/O latency and network delay). As shown in Fig. 2.2(a), this logical ring defines an order of preference between the primary and replica nodes. Caching is performed using the local SSD with a copy replicated to another node in the cluster. Each node consists of two neighbors, storing replicas on both/one of them. The node walks in the ring until it can find a replica to use if unsuccessful during the process of building the ring cluster. Once it has a replica, it can begin to write caching independently of what the other nodes are doing.

2.2. Network Approach. As a "linear" approach, the "Ring" structure has a drawback during searching and building replicas, since it has only one or two directions (e.g., previous and next neighbors). In order to improve system robustness and flexibility, we further proposed the "network" approach – a symmetric or asymmetric network, see Fig. 2.2(b), which is based on each node's preference ranking list of all its connected nodes (i.e., not limited to two nodes).

TABLE 2.1
Example of the "distance matrix" used in Network approach, which shows the ranking of each path.

From \ To	1	2	3	4	5	6	7	8
1	-	1	3	-	-	4	-	2
2	...	-	1	...	2	...	...	...
3	3	...	-	...	1	...	2	...
...	...	...	...	...	...	...	...	...
8	...	...	...	2	...	...	1	-

In real implementation, we introduce a "distance matrix" (an example is shown in Table 2.1) to maintain each node's preference list ranked by a customized "score" calculated based on multiple parameters such as network delay, I/O access speed, space/throughput utilization ratio, etc. This matrix is periodically updated through runtime measurement (e.g., heartbeat). For example, in Table 2.1, node 1's first neighbor is node 2, and its second neighbor is node 8, etc.

The main procedure of how to assign the replica nodes for each node is as following: Each node searches the matrix and selects its "closest" node as its replica node if possible. To avoid the "starvation" case where lots of nodes are choosing one single node or a small range of nodes as their replica nodes, AutoReplica also limits the maximum replica number per node. Lastly, each node can also have more than one replica node.

2.3. Multiple-SLA Network Approach. In real environment, rather than treating different nodes equally, the CPS system administrator is often required to differentiate the quality of service based CPS network SLAs and even workload characteristics. To support this requirement, we further develop the "multiple-SLA network" approach to allow each node to have more than one replica node with different configurations based on a replica configuration decision table.

TABLE 2.2
Replica configuration table for multiple SLAs.

Case	Workload		Destination				#Reps.
	SLA	Temp.	SSD_P	SSD_{R1}	SSD_{R2}	HDD_P	
1		✓	✓	✓			1
2			✓		✓		1
3	✓	✓	✓	✓	(✓)		1(2)
4	✓		✓	✓		(✓)	1(2)

An example of replica configuration table is shown in Table 2.2, where SSD_P, SSD_{R1}, SSD_{R2} and HDD_P stand for the SSD tier of the primary node, the SSD tier of the first replica node, the SSD tier of the second replica node, and the HDD tier of the primary node, respectively. It also considers:

- **SLA:** Related with importance of each node. Multiple SLAs is supported by utilizing multiple replica configurations. Although our example has only two degrees: "important" and "not important", AutoReplica supports more fine-grained degrees (even online-varying) SLAs.
- **Temperature:** Similar to [18], we use "data temperature" as an indicator to classify data into two categories according to their access frequency: "hot data" has a frequent access pattern, and "cold data" is occasionally queried.

Local HDD ($prmyNode.HDD$) can also be used as a replica destination (case 4 in table 2.2), and AutoReplica needs reduce the priorities of those write-to-HDD replica operations in order not to affect those SSD-to-HDD write back and HDD-to-SDD fetch operations in the I/O path. Additionally, techniques [19, 20, 21] are adopted to further improve the performance of the write-to-HDD queue in the I/O path considering multiple SLAs.

3. Cache and Replacement Policies. To maximize the I/O performance, AutoReplica uses *write back cache policy*. In detail, when the SSD tier (i.e., cache) is full, SSD-to-HDD eviction operations will be triggered in the (local) primary node ($prmyNode$); while in the replica node ($repNode$), the corresponding dataset will simply be removed from the $repNode.SSD$ without any additional I/O operations to the $repNode.HDD$. Alg. 3.1 shows a two-replica-node implementation. In fact, it can have any number of SSD replica nodes to support more fine-grained SLAs. Specifically, AutoReplica's cache and replacement policy is basically switching between two modes, namely "runtime mode" (line 19) and "online migration mode" (line 3 to 11) by periodically checks the $migTrigger$ condition (which considers runtime states such as load balancing and bandwidth utilization). If $migTrigger$ returns true, AutoReplica will select the "$overheat$" replica node (line 6) and is replaced with the next available replica node (line 7). After that, AutoReplica begins to run under the "migration mode" (line 14). If the "migrate out" replica node ($repNodeOut$) has no more "out-of-date" replica datasets (i.e., the migration is finished), AutoReplica then stops the migration by setting $migModeFlag$ to $false$ (line 16), and goes back to the runtime mode (line 19). We describe the details of the runtime mode and the migration mode cache policy in Sect. 3.1 and 3.2.

3.1. Runtime Mode Cache Policy. Under the runtime mode, AutoReplica searches the new I/O request in the local SSD cache partition (i.e., $prmyNode.SSD$). If it returns a cache hit, then AutoReplica either fetches it from the $prmyNode.SSD$ for a read I/O, or updates the new data to its existing cached copies in $prmyNode.SSD$ and the SSD replica partition in corresponding replica node(s) for a write I/O. For the cache miss case, AutoReplica first selects a victim to evict from the $prmyNode.SSD$ and all the victim's copies from $repNode.SSD$(s), and then AutoReplica only writes those unsync (with "dirty" flag) evicted datasets into $prmyNode.HDD$. AutoReplica then inserts the new dataset into both $prmyNode$ and all its $repNode.SSD$(s). If it is a read I/O, AutoReplica fetches it from $prmyNode.HDD$ to SSDs of the $prmyNode.HDD$ and also

Main Procedure of AutoReplica's Cache Policy

Note: (1) $prmyNode$ is the primary node. $repNodeRem$ and $repNodeOut$ are the original two replica node. $repNodeIn$ is the destination of migration which replaces $replicaNodeOut$.
(2) $write(inputData, inputDirtyFlag)$: A function that writes the *inputData* into the device. If the device is "SSD", then this function sets *dirtyFlag* of the *inputData* as *True*. If the device is "HDD", then $inputDirtyFlag$ can be ignored.
(3) $migrateTrigger()$: A function returns *True* if the subsystem needs to migrate due to imbalance load.
(4) T_W: window size (i.e., frequency) of migration condition checking.

```
Procedure cache (prmyNode, repNode1, repNode2)
1   migModeFlag = Flase
2   for each new I/O request newData ∈ IOStream on prmyNode do
3       /* check load balance */
4       if currTime mod T_W == 0 and migModeFlag ≠ True and migTrigger() ≠ False then
5           migModeFlag = True
6           repNodeOut = selectOverheatNode(repNode1, repNode2)
7           repNodeIn = selectNextReplica()
8           if repNodeIn == repNode1 then
9               repNodeRem = repNode2
10          else
11              repNodeRem = repNode1
12      /* online migrate mode cache policy */
13      if migModeFlag == True then
14          cacheOnlineMigrateMode(newData, prmyNode, repNodeRem, repNodeOut, repNodeIn)
15          if repNodeOut.SSD.sizeOfRepForPrmy() == 0 then
16              migModeFlag = False
17      /* runtime mode cache policy */
18      else
19          cacheRuntimeMode(newData, prmyNode, repNode1, repNode2)
20  return
```

FIG. 3.1. *Main procedure of AutoReplica's cache policy.*

send it to all its $repNode(s)$. It finally returns the fetched $cacheData$ to the user buffer in the memory. If it is a write I/O, AutoReplica simply writes it to SSDs of $prmyNode$ and all its $repNode$(s) with $dirtyFlag$ as "dirty", since it is unsync new data. An example is shown in Fig. 3.2, where for the write I/O data "F", AutoReplica first writes back a victim "E" from $prmyNode.SSD$ to $prmyNode.HDD$, and deletes "E" from both $prmyNode.SSD$ and $reNode.SSD$. Lastly, it writes "F" to both $prmyNode.SSD$ and $reNode.SSD$. Our implementation is also compatible for other replacement algorithms to implement the victim selection function, such as Glb-VFRM [18] and GREM [22].

3.2. Online Migration Mode Cache Policy. AutoReplica uses a cost-efficient migrate-on-write scheme called "fusion cache" to migrate replicas from one $repNode$ to the other. The main idea is that instead of pausing the subsystem and copying all existing replicas from the old node ($repNodeOut$) to the new node ($repNodeIn$), regardless of whether these data pieces are necessary or not, "fusion cache" keeps the subsystem alive and only writes new incoming datasets to $repNodeIn$ and keeps those "unchanged" cached data on $repNodeOut$. Eventually, $repNodeIn$ will replace $repNodeOut$. In other words, AutoReplica mirrors the $prmyNode.SSD$ by using the *unibody* of $repNodeOut$ and $repNodeIn$ to save lots of bandwidth.

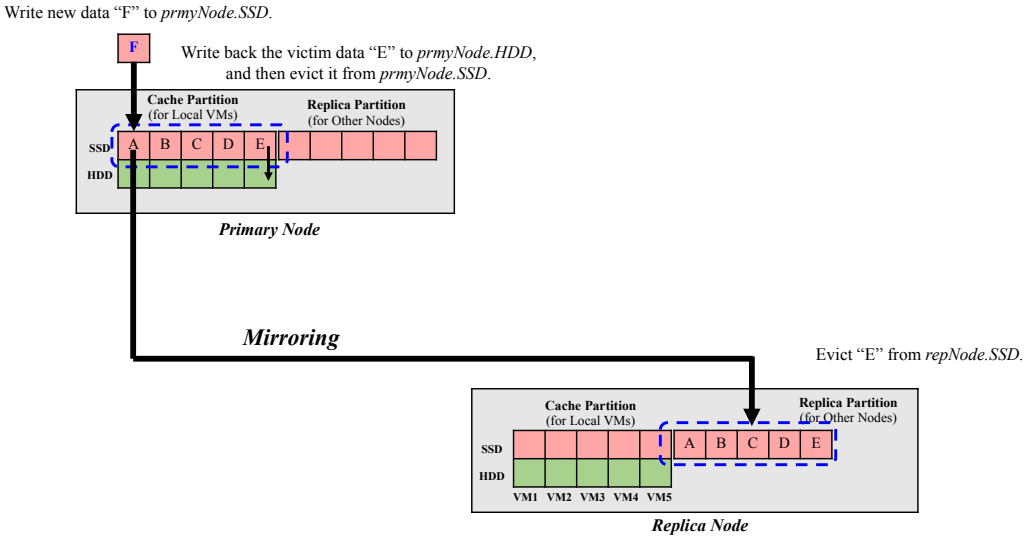


FIG. 3.2. *Example of runtime cache policy.*

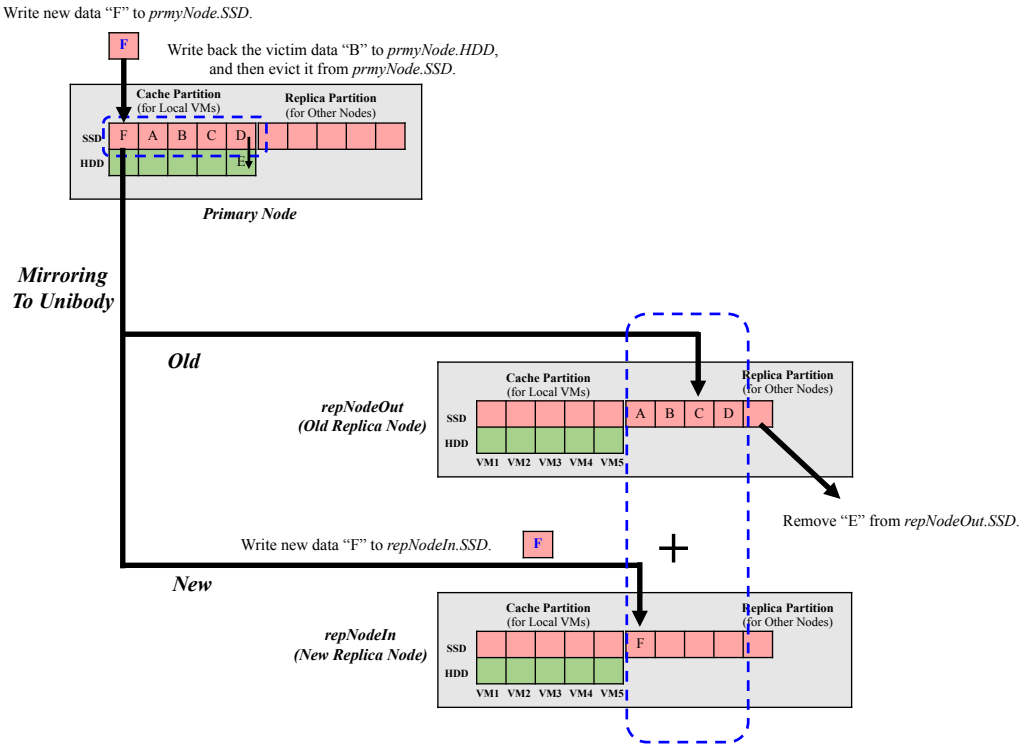


FIG. 3.3. *Example of online migration cache policy.*

An example is depicted in Fig. 3.3, where only one replica node is needed to be "migrated out" and one new replica node is needed to take over those cached datasets. When a new write I/O data "F" comes to the *prmyNode*, similarly, AutoReplica first writes back the victim "E" from $prmyNode.SSD$ to $prmyNode.HDD$ since the cache is full. Then, "E" on the old replica node is directly deleted. After that, the new write I/O "F" will be inserted to the $prmyNode.SSD$ and its new $repNodeIn.SSD$. As mentioned, the *prmyNode* (e.g., the

blue dash box on top of Fig. 3.3) is mirrored in the "fusion cache" across the old and new *repNodes* (e.g., the blue dash box in on the right side of Fig. 3.3). Notice that if "F" is a read I/O, AutoReplica will not migrate it from the old to the new replica node.

To sum up, unlike the traditional pro-active migration, AutoReplica is following this "migrate on write" scheme which is lazy and cost-efficient.

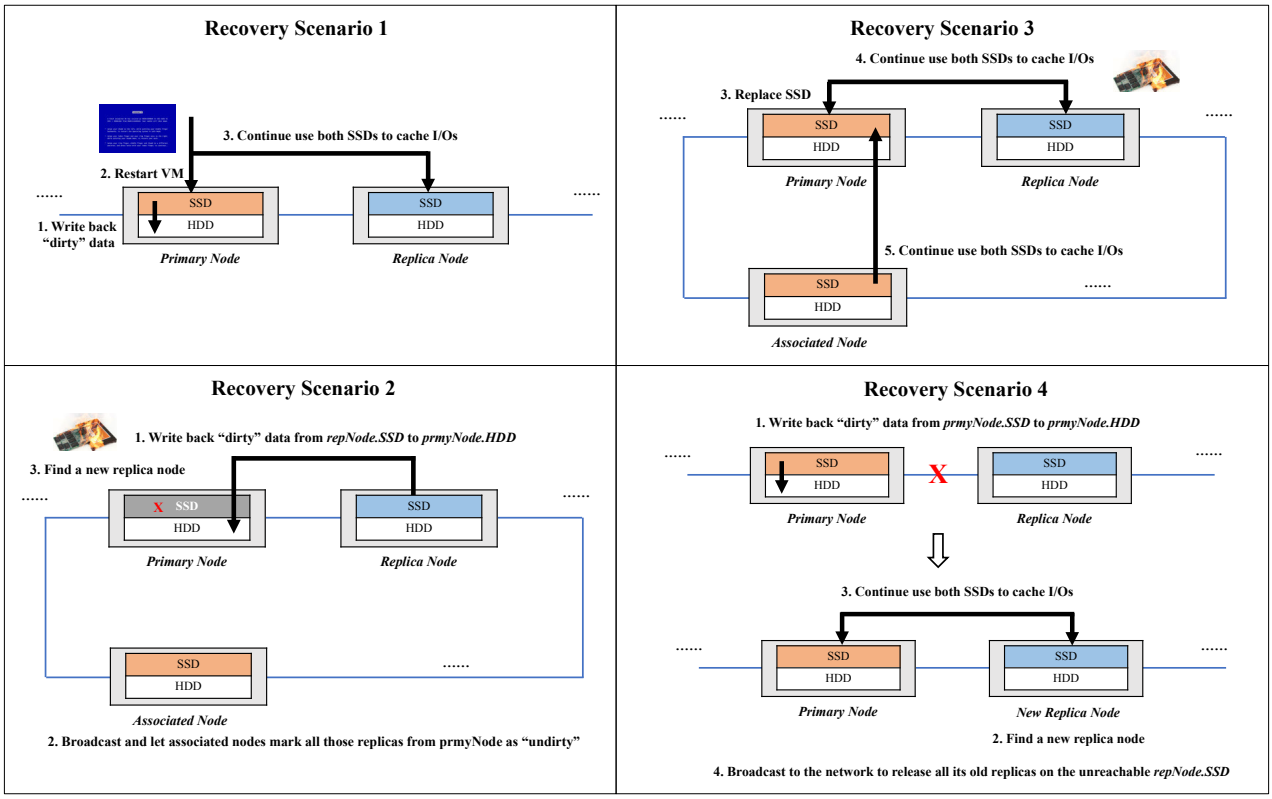


FIG. 4.1. *Example of AutoReplicas recovery scenario.*

4. Recovery Policy. AutoReplica maintains additional replicas in the remote SSDs to prepare for recoveries for different failures. Specifically, it has different procedures to recover from failures covering the following four scenarios:

4.1. VM Crash on Primary Node. A very common failure is that a CPS service virtual machine crashes on the primary node. As shown Fig. 4.1(1) and Fig. 4.2, AutoReplica first closes out the VMDK. It then writes back "dirty" datasets from $prmyNode.SSD$ to $prmyNode.HDD$, and marks all $prmyNode$'s replicas in $repNode.SSD(s)$ with "nondirty" flag. After that, it restarts crashed VM on $prmyNode$, and continues to forward incoming I/O requests on both $prmyNode.SSD$ and $repNode.SSD$.

4.2. Primary Node Cache Device Failure. A primary node cache storage device failure will result in its inability to continue to write caching. As shown in Fig. 4.1(2) and Fig. 4.3, AutoReplica first writes back "dirty" datasets from $repNode.SSD$ to $prmyNode.HDD$, and marks all $prmyNode.SSD$'s replicas on $repNode.SSD(s)$ with "nondirty" flag. It then broadcasts this "unavailable" information to notify those nodes having replicas of this failure $prmyNode$ (called "associated nodes") to write back "dirty" datasets from their own SSD to HDD. Notice that replicated datasets with "nondirty" flags are still kept in the $repNode.SSD$(s). AutoReplica further finds and replaces the SSD on $prmyNode$. After that, it continues to write incoming I/O requests on both $prmyNode.SSD$ and $repNode.SSD$. It also continues to let those "associated nodes" to write new replicas to $prmyNode.SSD$.

[Recovery Scenario 1] VM Crash on Primary Node	
	Procedure $recoverFromPrmyNodeVMCrash()$
1	Write back "dirty" data from $prmyNode.SSD$ to $prmyNode.HDD$, and mark their copies in $repNode.SSD$ with "nondirty" flag.
2	Restart crashed VM on $prmyNode$.
3	Continue to forward incoming I/O requests on both $prmyNode.SSD$ and $repNode.SSD$.
4	**return**

FIG. 4.2. *Main procedure of recovery scenario 1: VM crash on primary node.*

[Recovery Scenario 2] Primary Node Cache Device Failure	
	Procedure $recoverFromPrmyNodeSSDFailure()$
1	Write back "dirty" data from $repNode.SSD$ to $prmyNode.HDD$, and mark their copies in $repNode.SSD$ with "nondirty" flag.
2	Broadcast this "unavailable" information to the network to let those nodes having replicas in this failure $prmyNode$ ("associated nodes") to write back "dirty" data from their own SSD to HDD, and keep copies with "nondirty" flag in those associated nodes.
3	Replace SSD on $prmyNode$.
4	Continue to write incoming I/O requests on both $prmyNode.SSD$ and $repNode.SSD$.
5	Continue to let those "associated nodes" to write new replicas to $prmyNode.SSD$.
6	**return**

FIG. 4.3. *Main procedure of recovery scenario 2: Primary node cache device failure.*

4.3. Replica Node Cache Device Failure. When a replica node detects a cache device (i.e., SSD) failure, it will disconnect from the primary node and reject any future connection attempts from that node with an error response. As shown in Fig. 4.1(3) and Fig. 4.4, AutoReplica then writes back "dirty" dataset from $prmyNode.SSD$ to $prmyNode.HDD$, but still marks and keeps them in $prmyNode.SSD$ with "nondirty" flag. After a new replica node is found by using dynamic evaluation process, AutoReplica continues to write incoming I/O requests on both $prmyNode.SSD$ and new $repNode.SSD$. Notice that policy in Sect. 4.2 takes responsibility for recovering this failure device.

[Recovery Scenario 3] Replica Node Cache Device Failure	
	Procedure $recoverFromRepSSDFailure()$
1	Write back "dirty" data from $prmyNode.SSD$ to $prmyNode.HDD$, and mark their copies in $prmyNode.SSD$ with "nondirty" flag.
2	Find a new replica node by using dynamic evaluation process.
3	Continue to write incoming I/O requests on both $prmyNode.SSD$ and new $repNode.SSD$.
4	**return**

FIG. 4.4. *Main procedure of recovery scenario 3: Replica node cache device failure.*

4.4. Communication Failure Between Primary & Replica Node. When the primary node detects a non-recoverable communication failure between the primary and replica hosts, AutoReplica will be unable to continue to write caching. To recover from this failure, as shown in Fig. 4.1(4) and Fig. 4.5, AutoReplica daemon will write back "dirty" data from $prmyNode.SSD$ to $prmyNode.HDD$ to ensure all cached data are

updated to the backend HDD. Next, it will start the dynamic evaluation process to find a new replica node to replace the unreachable replica node. It will then continue to use both SSDs to cache I/Os following the "fusion cache" design in migration policy. Finally, it will broadcast to the network to release all its old replicas on the unreachable $repNode.SSD$(s).

[Recovery Scenario 4] Communication Failure between Primary and Replica Node	
	Procedure *recoverFromCommFailure*()
1	Write back "dirty" data from *prmyNode.SSD* to *prmyNode.HDD*, and mark their copies in *prmyNode.SSD* with "nondirty" flag.
2	Find a new replica node by using dynamic evaluation process.
3	Continue to write incoming I/O requests on both primary SSD and new replica SSD.
4	Broadcast to the network to release all its old replicas on the unreachable *repNode.SSD*.
5	**return**

FIG. 4.5. *Main procedure of recovery scenario 4: Communication failure between primary and replica node.*

5. Parallel Prefetching. Lastly, replicates can also be used to enable parallel prefetching from multiple nodes (similar like parallel stripping in RAID [23, 24]), especially for read operations. An example is shown in Fig. 5.1, where we split the dataset (with size of C) to prefetch (e.g., a file) into two parts (with sizes of αC and C), and load each part from the primary and replica node. Assume the access speed of $prmyNode.SSD$ is λ_1 (GB/Sec) and the access speed of $repNode.SSD$ (including the network delay) is λ_2 (GB/Sec). Since the main target for parallel prefetching is to reduce the total I/O time, i.e., makespan of each I/O request, we can convert our problem into the following optimization framework:

Optimize:

$$\max\left(\frac{\alpha C}{\lambda_1}, \frac{(1-\alpha)C}{\lambda_2}\right) \tag{5.1}$$

Subject to:

$$\alpha \in [0,1] \tag{5.2}$$

$$\lambda_1 \geq \lambda_2 > 0 \tag{5.3}$$

$$\frac{C}{\lambda_1} \geq \max\left(\frac{\alpha C}{\lambda_1}, \frac{(1-\alpha)C}{\lambda_2}\right) \tag{5.4}$$

Eq. 5.1 is the objective function which minimizes the overall makespan of an I/O request. The makespan is determined by the maximum value of the I/O operating time of each side (i.e., $prmyNode.SSD$ and $repNode.SSD$). Eq. 5.2 ensures that the branching ratio of two streams should be meaningful. Eq. 5.3 reflects that the local SSD I/O speed (i.e., from $prmyNode.SSD$) is usually greater than remote (i.e., $repNode$) I/O speed including network delay. Notice that this constraint can be relaxed as "$\lambda_1 > 0$ and $\lambda_2 > 0$", if the remote I/O speed is higher (which is true in some rare cases), but the optimization framework remains the same. Eq. 5.4 further ensures that the parallel prefetching operation should only be triggered when it *can* help to reduce the I/O makespan. Based on this result, Fig. 5.3 further shows the example of the decision maker workflow for parallel prefetch, where the "StatusMonitor" reports to "PararllelPrefectDeamon" and the latter component switches between the "ParallelPrefetchMode" and "LocalPrefetchMode".

We then plot these functions and constraints into Fig. 5.2, where the red line is the objective function curve, and the blue line is the constraint of Eq. 5.4. We can see that there exists a minimum point at the cross point of $f(\alpha) = \frac{(1-\alpha)C}{\lambda_2}$ and $g(\alpha) = \frac{\alpha C}{\lambda_1}$. In order to calculate this sweet spot, we let:

$$\frac{\alpha C}{\lambda_1} = \frac{(1-\alpha)C}{\lambda_2} \tag{5.5}$$

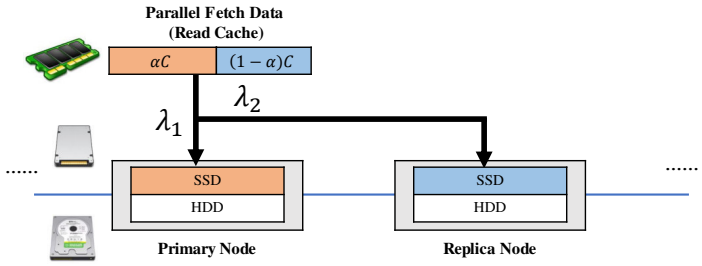


FIG. 5.1. *Example of parallel prefetch enabled read cache.*

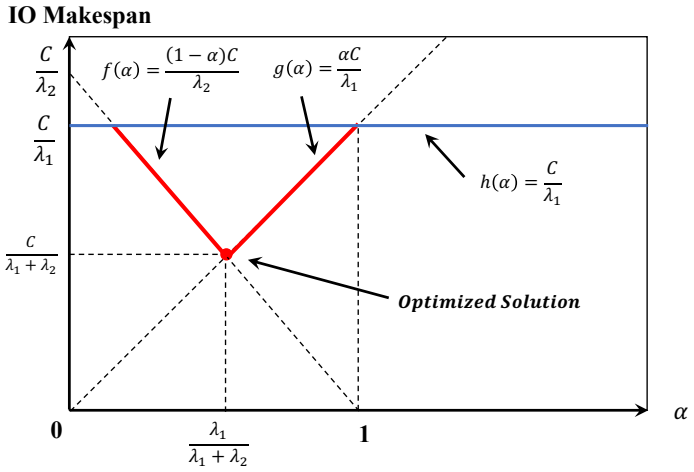


FIG. 5.2. *Finding the optimized solution of prefetching stream division.*

Then, we can get the minimum makespan ($\frac{\alpha C}{\lambda_1+\lambda_2}$) when:

$$\alpha = \frac{\lambda_1}{\lambda_1 + \lambda_2} \tag{5.6}$$

We can easily extend it to support any number of replica nodes. Alg. 5.4 first describes the main procedure of parallel fetching daemon, which triggers the parallel prefetching by periodically checking whether the access speed of its all $repNode.SSD(s)$ (including the network delay) is close enough to the access speed of local $prmyNode.SSD$ (by comparing their difference with a preset threshold ε), and the current utilization ratio of throughput of the $repNode.SSD(s)$ is less than a threshold Thr. Notice that the time window T_W does not necessarily to be same as the sliding window previously mentioned in Alg. 3.1. The "ParallelPrefetchMode" will be triggered if all these conditions are satisfied, and the parallel fetching policy then calculates and assigns the branching ratio of dataset to be loaded from each node. Otherwise, AutoReplica will keep running "LocalPrefetchMode", which only fetches data from the local $prmyNode.SSD$.

6. Evaluation. In this section, we investigate the performance of our proposed AutoReplica solution under different CPS use cases.

6.1. Implementation Configurations. Table 6.1 summarizes the configuration of our CPS server testbed. Figure 6.1 also illustrates the architecture of our VMware-based implementation. Specifically, there are multiple CPS server nodes in the cluster. Each of these CPS server nodes runs the VMware ESXi hypervisor [17] to host multiple VMs that are working for nearby CPS devices based on location distribution. AutoReplica works on the VMware's ESXi in the "user mode". More accurately, with some internal customization, AutoReplica is

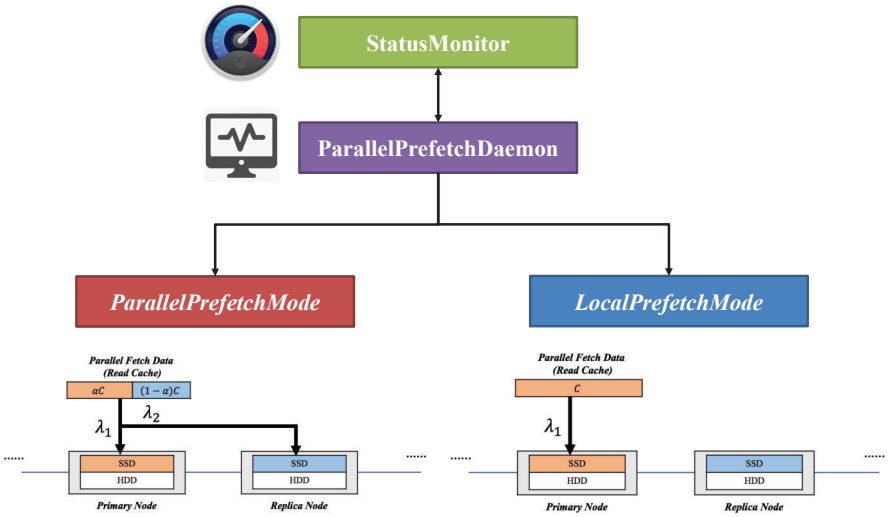


FIG. 5.3. *Decision maker for parallel prefetch.*

very close to a "pseudo-kernel mode" application. AutoReplica consists of 4 components: a vSphere web client plug-in, a CIM provider, multiple I/O filter library instances and a daemon process on each VM. CPS server nodes are connected by high-speed fiber channels, and due to the advantage of SSD and fiber channels, remote SSDs access speed can be faster than local HDDs speed. AutoReplica strives to cache hot datasets onto the SSD tier and bypass those cold datasets onto the HDD tier.

TABLE 6.1
Configuration of a single CPS server.

Component	Specs
Server	PowerEdge R630 Server
Processor	Intel(R) Xeon(R) CPU E5-2660 v4
Processor Speed	2.00GHz
Processor Cores	56 Cores
Processor Cache Size	35M
Memory Capacity	64GB RDIMM
Memory Data Rate	2400 MT/s
Operating System	Ubuntu 14.04 LTS
Docker Version	17.03
VM Hypervisor	VMware Workstation 12.5

6.2. Performance Metrics. In our evaluation, we mainly focus on the following four metrics:

- **Total recovery time:** the cumulative recovery time of all CPS server nodes.
- **Coefficient variation (C.V.) of total recovery time:** balance degree of the cumulative recovery time across all CPS server nodes. The less C.V. is, the better balance is achieved.
- **Overhead:** Total extra I/O and network traffic for recovery.
- **Coefficient variation (C.V.) of overhead:** balance degree of the overhead across all CPS server nodes. The less C.V. is, the better balance is achieved.

We use the open source measurement tools (e.g., `dstat` [25], `iostat` [26], `blktrace` [27]) to measure performance metrics such as total recovery time, disk I/O rate, and network traffic. We evaluate the recovery correctness and efficiency of different failure scenarios that are manually triggered and drawn from well-tuned temporal interval distributions. We also evaluate the total recovery time and extra I/O overhead under different cluster sizes. Different SLA configurations are further considered to evaluate the benefit and overhead of multiple

Main Procedure of Parallel Fetching Daemon

Note: (1) $currUtil_{IOPS}$: current throughput utilization rate of a node.
(2) ϵ: preset threshold to compare the difference between local and remote access speed.
(3) Thr_{IOPS}: preset threshold of upper bound of throughput utilization rate of a node to be perfected from.

Procedure $parallelFetchDaemon()$
1 $parallelReadFlag = False$
2 **while** $True$ **do**
if $(currTime \bmod T_W == 0)$ **and** $(\lambda(prmyNode.SSD) - \sum_{i \in RepNodes} \lambda(repNode(i).SSD) \leq \epsilon)$ **and**
3 $(repNode(s).currUtil_{IOPS} \leq Thr_{IOPS})$ **then**
4 $parallelReadFlag = True$
5 **else**
6 $parallelReadFlag = False$
7 **return**

Parallel Fetching Policy

Note: (1) $\lambda(prmyNode.SSD)$: the IO speed of $prmyNode.SSD$.
(2) $\lambda(repNode(i).SSD)$: the end-to-end IO speed of $repNode(i).SSD$ (including network delay).

Procedure $parallelFetch(data)$
1 **if** $parallelReadFlag == True$ **then**
2 fetch size of $\left[\frac{\lambda(prmyNode.SSD)}{\lambda(prmyNode.SSD)+\sum_{i \in RepNodes} \lambda(repNode(i).SSD)} \cdot |data|\right]$ data from $prmyNode.SSD$
3 fetch size of $\left[\frac{\lambda(prmyNode(i).SSD)}{\lambda(prmyNode.SSD)+\sum_{i \in RepNodes} \lambda(repNode(i).SSD)} \cdot |data|\right]$ data from $repNode(i).SSD$
4 **return**

FIG. 5.4. *Parallel prefetching procedure.*

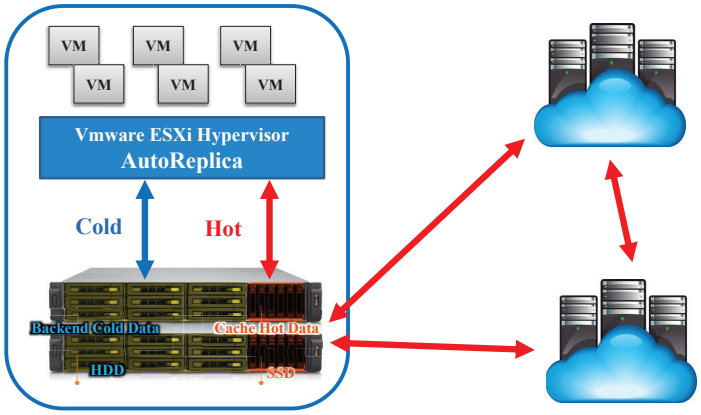


FIG. 6.1. *Architecture of AutoReplica implementation.*

replicas.

6.3. Comparison Solutions. To conduct fair comparison, we implemented the following three representative solutions for the CPS cluster:

- `NoReplication`: No replications will be generated and maintained in the CPS cluster. Once a CPS server node is failed, data cached in the SSD cannot be recovered from other nodes or local HDD tier. As a result, re-collection and re-computation from CPS devices are triggered.

- `Replication(IM)`: A pro-active replication solution, which conducts "Immediate Migration (IM)" on recovery. To conduct fair comparison, we maintain the same replication assignment as AutoReplica [28] for this solution.
- `AutoReplica(MOW&PF)`: Our proposed AutoReplica solution which has fusion cache with migrate-on-write (MOW), and parallel prefetching (PF) features enabled.

Meanwhile, to consider more scenarios in reality, we also evaluate both homogeneous and heterogeneous CPS VM servers and hardware. Details will be discussed in the following two subsections.

6.4. Study on Homogeneous CPS Clusters. We first investigate the homogeneous CPS cluster use case, which is often referred to as a "symmetrical" CPS structure where same CPS devices are widely deployed in the cluster. This use case is popular due to its simplicity and system consistency [29].

Table 6.2 shows the statistics of the experiment configurations. In detail, $clusterSize$ is iterated from 10 to 100 CPS server nodes. We use $avgRepNode\#$ and $avgUsrNode\#$ to demote the average number of replication nodes that each node has and the average number of replicated remote node copies that each node is hosting, respectively. Additionally, $connectRatio$ shows that the connectivity of the cluster, which equals the ratio of the number of paths between two nodes to the upper bound of all possible paths (e.g., for N-node cluster, at most, we can have $\frac{N^2}{2}$ paths). The larger $connectRatio$ is, the higher chance a node will be recovered from others. We further use $avgSSDBW$ and $avgHDDBW$ to denote the average bandwidth of each node at the SSD tier and the HDD tier, respectively. Finally, $readIO\%$ gives the read I/O ratio of each CPS server node. Notice that for the homogeneous use case, both hardware-related (e.g., $avgSSDBW$ and $avgHDDBW$) and workload-related (e.g., $readIO\%$) factors are the same among different cluster size cases.

Table 6.2
Statistics of configurations of homogeneous CPS cluster.

clusterSize	10	20	30	40	50	60	70	80	90	100
avgRepNode#	2.20	2.45	2.63	2.33	2.62	2.45	2.41	2.39	2.43	2.45
avgUsrNode#	1.70	2.00	2.03	1.95	2.04	1.98	2.00	1.93	1.91	1.91
connectRatio(%)	48.00	37.00	48.67	47.13	47.60	48.22	51.06	50.13	49.04	49.96
avgSSDBW(TB/hour)	36.00	36.00	36.00	36.00	36.00	36.00	36.00	36.00	36.00	36.00
avgHDDBW(TB/hour)	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58	0.58
readIO(%)	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00	30.00

Figs. 6.2 to 6.5 depict the results of 10 homogeneous CPS cluster use cases, where both device failure and communication failure rates are following the uniform distribution. As shown in Fig. 6.2, `NoReplication` has long total recovery time, because that once a device failure occurs, `NoReplication` has no backups and has to request CPS devices to collect and send data to it again, and it further needs to re-compute the lost data. Meanwhile, `AutoReplica` saves 9.28% of total recovery time from the `Replication` case. The reason is that `AutoReplica` lazily recoveries from its neighbors using the migration-on-write technique and maximizes I/O bandwidth using the parallel prefetching technique. In contrast, `Replication` has to pause and migrate everything immediately once a device or communication failure happens.

We further evaluate the coefficient variation of total recovery time in Fig. 6.3, where the recovery time of `Replication` has a higher chance to be unbalanced among nodes, while our `AutoReplica` has similar balancing degree to `NoReplication`. This result is promising since in a homogeneous use case, more balanced total recovery distribution helps to make the cluster more robust and stable.

We next investigate the extra I/O and network traffic for recovery. As shown in Fig. 6.4, `Replication` has very large overhead, because it has to migrate everything immediately once a failure happens. Meanwhile, even with the need for maintaining additional replications, `AutoReplica` still has lower overhead than all `NoReplication` cases. The main reason behind it is that once a device failure occurs, `NoReplication` has no backups and has to request CPS devices to send data to it again, and it also has to recompute the lost data. These operations trigger a huge amount of extra I/O and network bandwidth consumption.

We further check the coefficient variation of overhead in Fig. 6.5. Neither `Replication` nor `AutoReplica` are load balanced as the `NoReplication` case. Additionally, we observe that although `AutoReplica` saves lots

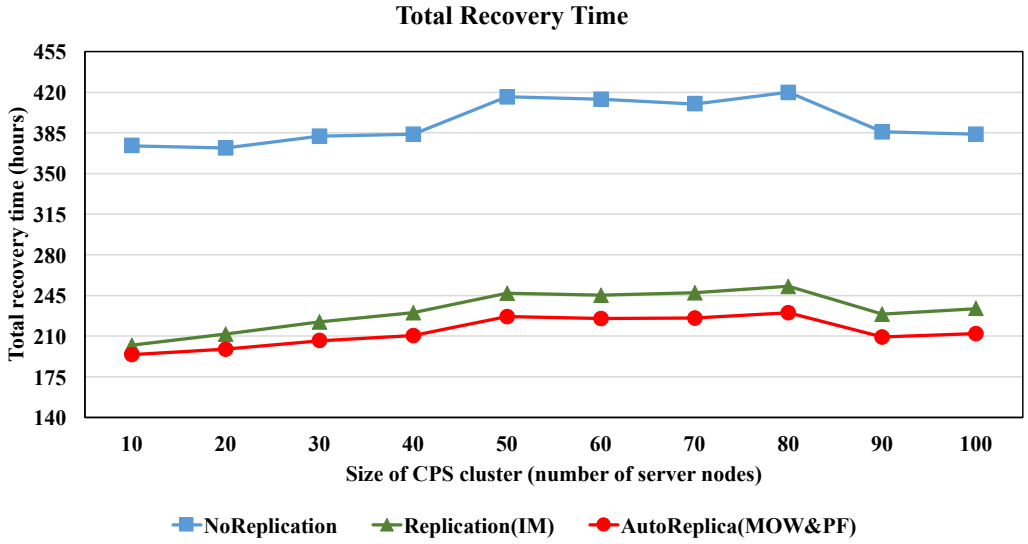


FIG. 6.2. *Total recovery time under different homogeneous CPS cluster sizes.*

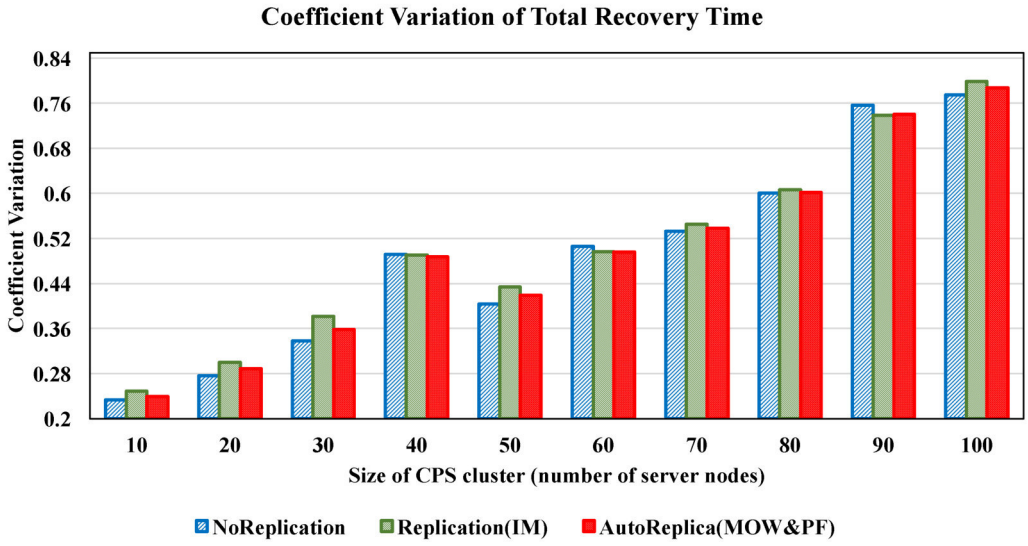


FIG. 6.3. *Coefficient variation of total recovery time under different homogeneous CPS cluster sizes.*

of I/O and network bandwidth, the difference of the overhead balancing degree between `AutoReplica` and `Replication` cases is very slight. The reason is mainly that we deployed the same replication assignment and failure distribution in these two cases in order to conduct fair comparison.

6.5. Study on Heterogeneous CPS Clusters. We further conduct a set of heterogeneous CPS experiments. Table 6.3 shows the configuration of heterogeneous CPS clusters. Hardware factors (such as $avgSSDBW$ and $avgHDDBW$) and workload factors (such as $readIO\%$) are varying among CPS server nodes. Notice that in terms of I/O pattern, CPS workloads usually are write-intensive [30, 31, 32, 33].

Similarly, as we can see from Fig. 6.6, `NoReplication` has the highest total recovery time due to its recomputation cost, and `AutoReplica` still performs better than `Replicaiton` for all cluster sizes. Notice that for cluster size of 20 and 60, we found the failure interval is relatively larger than that for other cluster sizes. As a result, the scenarios with cluster size of 20 and 60 have lower chances to have multiple failures happen

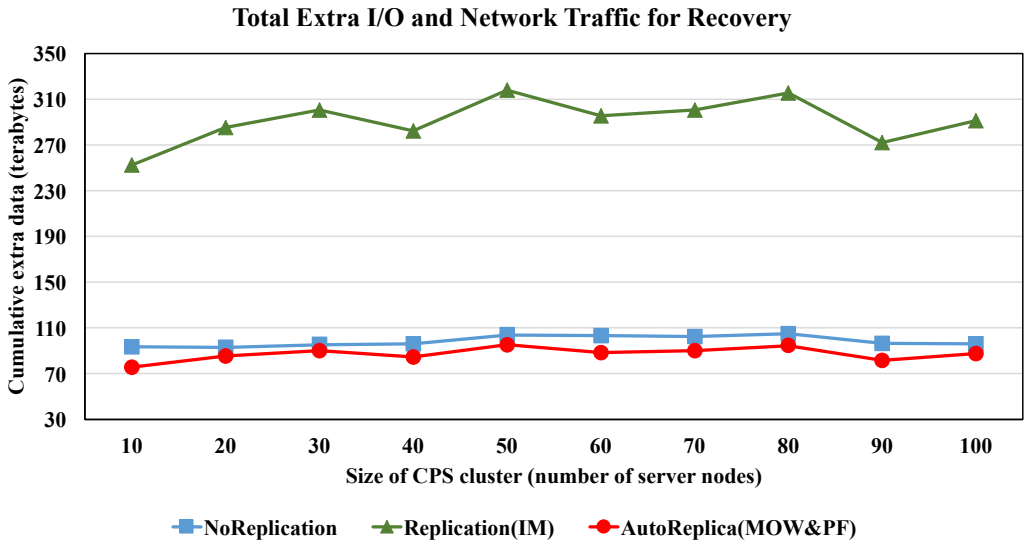


FIG. 6.4. *Total extra I/O and network traffic for recovery under different homogeneous CPS cluster sizes.*

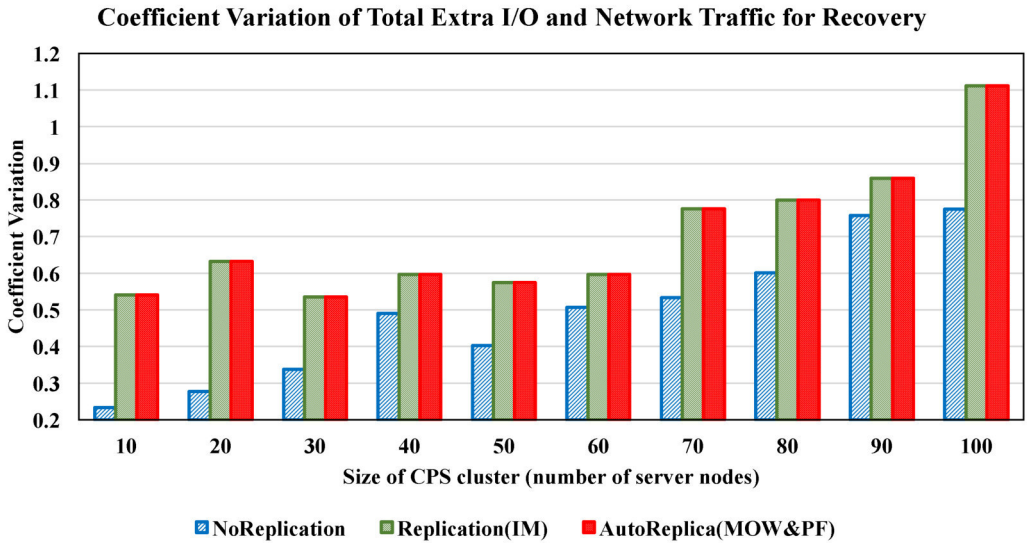


FIG. 6.5. *Coefficient variation of total extra I/O and network traffic for recovery under different homogeneous CPS cluster sizes.*

simultaneously, which then reduces the recovery congestion.

Fig. 6.7 reflects the total recovery time balancing degree. Unlike the homogeneous use case, `Replicaiton` and `AutoReplica` do not have the total recovery time as balanced as `NoReplication` under the heterogeneous use case. The reason is that `NoReplicaiton`'s recovery time is mainly dominated by the CPS device re-collecting and re-sending data. Hence, `NoReplicaiton`'s recovery time is highly coupled with the failure distribution. On the other hand, the recovery time of `Replicaiton` and `AutoReplica` is depending on both failure distribution and replication network assignment.

Similar to the homogeneous use case (e.g., Fig. 6.4), Fig. 6.8 shows that in the heterogeneous use case, `Replication` has the highest overhead. While, `AutoReplica` achieves the shortest total recovery time with a slightly higher overhead compared to `NoReplication`. Fig. 6.9 also show the results of overhead balancing as

TABLE 6.3
Statistics of configurations of heterogeneous CPS cluster.

clusterSize	10	20	30	40	50	60	70	80	90	100
avgRepNode#	2.40	2.45	2.77	2.58	2.60	2.45	2.39	2.38	2.44	2.58
avgUsrNode#	1.80	1.95	2.17	2.05	2.10	1.95	1.83	1.95	1.84	2.08
connectRatio(%)	52.00	46.00	49.56	48.88	49.04	48.00	47.47	49.59	48.62	49.66
avgSSDBW(TB/hour)	37.60	41.15	39.27	39.13	39.34	40.72	41.04	40.73	40.68	40.71
avgHDDBW(TB/hour)	0.50	0.50	0.51	0.51	0.50	0.51	0.50	0.50	0.49	0.50
readIO(%)	35.60	36.70	37.57	37.43	38.08	36.92	38.03	36.91	37.80	37.34

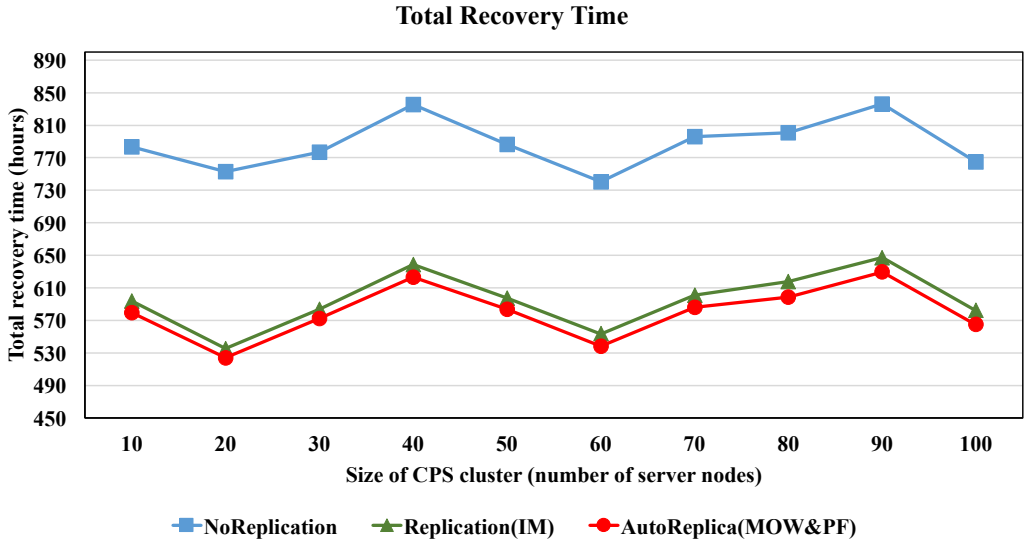


FIG. 6.6. *Total recovery time under different heterogeneous CPS cluster sizes.*

similar as the homogeneous case.

7. Related Work. Replications are widely used in the big data and cloud computing era [4, 34, 35, 36]. Replica's are useful in an incident of data loss for recovery [37, 7] and also for improving performance [38, 33]. Data loss can be incurred by many factors such as software-hardware failure, natural disaster at data center location, power surge etc. Moreover, when there are more highly-visited file gathered in some nodes with poor storage capacity, it will cause a hot issue which may reduce the overall performance of the system, so replication is used to guarantee performance in such critical situations.

Facebook's proprietary HDFS implementation [39] constrains the placement of replicas to smaller groups in order to protect against concurrent failures. MongoDB [28] is a NoSQL database system that uses replicas to protect data. Its recovery scheme is based on the election among live nodes. Copyset [40] is a general-purpose replication technique that reduces the frequency of data loss. [41] designed a novel distributed layered cache system built on the top of the Hadoop Distributed File System. Studies [42, 43, 44, 45, 46, 47, 48] investigated SSD and NVMe storage-related resource management problems, in order to reduce the total cost of ownership and increase the Flash device utilization to improve the overall I/O performance.

Replication creation strategies of based on the frequency of data operation [15, 49, 16] and file heat [50], is proposed to solve the problem of uneven distribution of data in auto-sharing and hybrid clouds. [51] focuses on the dynamic replica placement and selection strategies in the data grid environment. [52] proposed a replication strategy based on the access pattern of file in order to optimize load balancing for large-scale user access in cloud-based WebGISs. [53] highlighted the challenges involved in making a replica selection scheme explicitly cope with performance fluctuations in the system and environment. Replication can also help in reducing communication overhead among different nodes in cloud [54, 55, 56, 57, 58, 14].

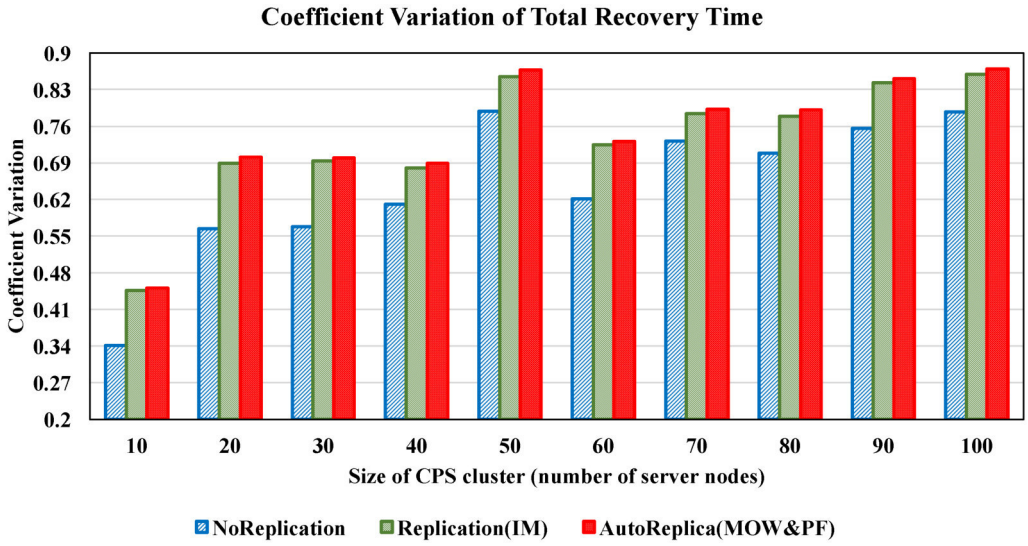


FIG. 6.7. *Coefficient variation of total recovery time for recovery under different heterogeneous CPS cluster sizes.*

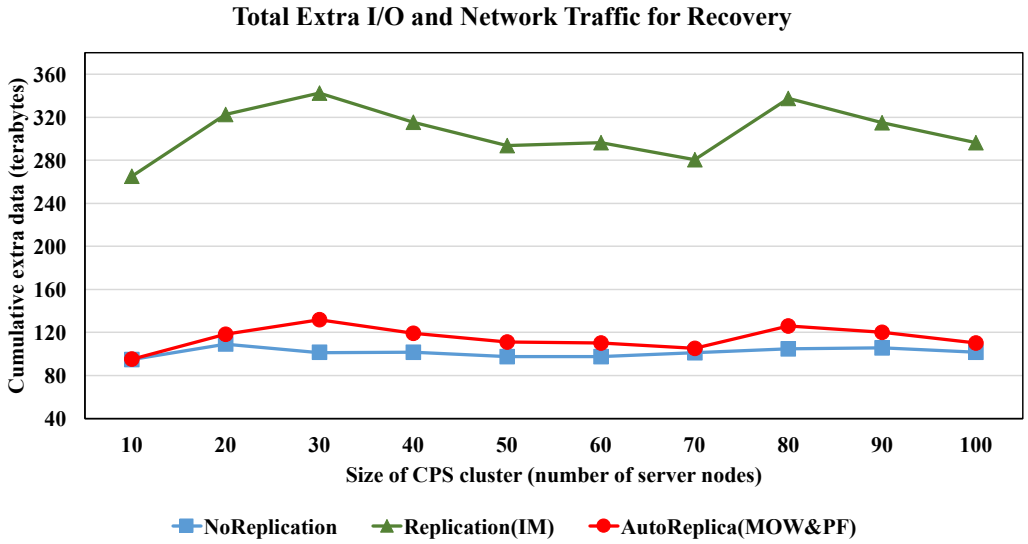


FIG. 6.8. *Total extra I/O and network traffic for recovery under different heterogeneous CPS cluster sizes.*

8. Conclusion. We proposed a complete data replica manager solution called "AutoReplica', working in distributed caching and data processing systems using SSD-HDD tier storages. AutoReplica balances the trade-off between the performance and fault tolerance by storing caches in replica nodes' SSDs. It has three approaches to build the replica cluster in order to support multiple SLAs, based on an abstract "distance matrix" which considers preset priorities, workload temperature, network delay, storage access latency, and etc. AutoReplica can automatically balance loads among nodes, and can conduct seamlessly online migration operation (i.e., migrate-on-write scheme), instead of pausing the subsystem and copying the entire dataset from one node to the other. AutoReplica further supports parallel prefetching from both primary node and replica node(s) with a new dynamic optimizing streaming technique to improve I/O performance. In the future, we plan to work on AutoReplica's compatibility with other hypervisors such as KVM/Xen and Virtual Box.

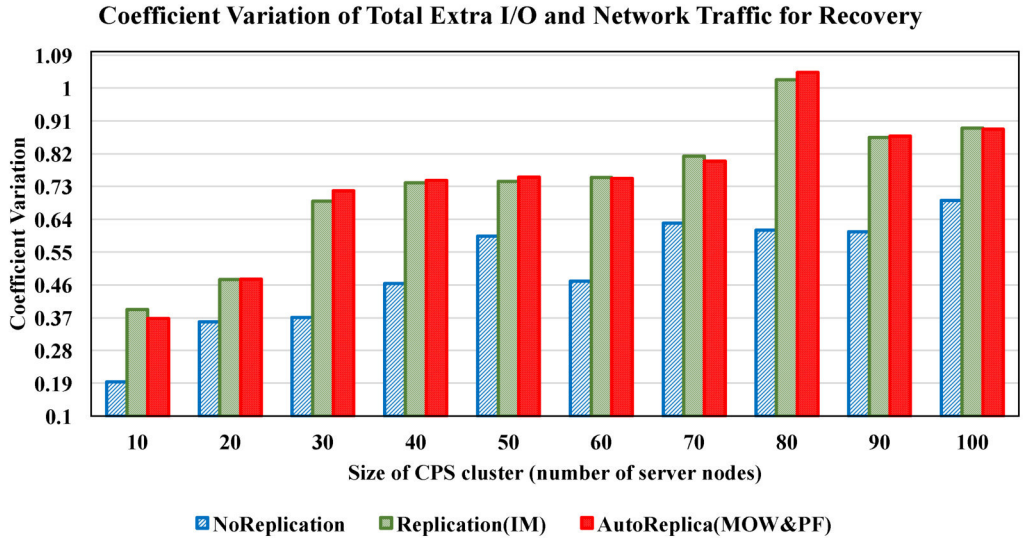


FIG. 6.9. *Coefficient variation of total extra I/O and network traffic for recovery under different heterogeneous CPS cluster sizes.*

Acknowledgment. This work was completed during Zhengyu Yang, Janki Bhimani and Jiayin Wang's internship at Storage Software Group and Performance and Datacenter Team, Memory Solution Lab, Samsung Semiconductor Inc. (CA, USA), and was partially supported by NSF grant CNS-1452751.

REFERENCES

[1] A. A. OMAR, A. GAWANMEH, AND A. APRIL, On the analysis of cyber physical systems, IN *Leadership, Innovation and Entrepreneurship as Driving Forces of the Global Economy*. SPRINGER, 2017, PP. 297–302.

[2] B. CHEN, Z. YANG, S. HUANG, X. DU, Z. CUI, J. BHIMANI, AND N. MI, Cyber-Physical System Enabled Nearby Traffic Flow Modelling for Autonomous Vehicles, IN *36th IEEE International Performance Computing and Communications Conference, Special Session on Cyber Physical Systems: Security, Computing, and Performance (IPCCC-CPS)*. IEEE, 2017.

[3] A. GAWANMEH AND A. ALOMARI, Challenges in formal methods for testing and verification of cloud computing systems, *Scalable Computing: Practice and Experience*, VOL. 16, NO. 3, PP. 321–332, 2015.

[4] B. A. MILANI AND N. J. NAVIMIPOUR, A comprehensive review of the data replication techniques in the cloud environments: Major trends and future directions, *Journal of Network and Computer Applications*, VOL. 64, PP. 229–238, 2016.

[5] Z. YANG AND D. EVANS, Automatic Data Placement Manager in Multi-Tier All-Flash Datacenter, PATENT US62/534 647, 2017.

[6] Z. YANG, M. HOSEINZADEH, A. ANDREWS, C. MAYERS, D. T. EVANS, R. T. BOLT, J. BHIMANI, N. MI, AND S. SWANSON, AutoTiering: Automatic Data Placement Manager in Multi-Tier All-Flash Datacenter, IN *36th IEEE International Performance Computing and Communications Conference*. IEEE, 2017.

[7] S. M. TONNI, M. Z. RAHMAN, S. PARVIN, AND A. GAWANMEH, Securing big data efficiently through microaggregation technique, IN *Distributed Computing Systems Workshops (ICDCSW), 2017 IEEE 37th International Conference on*. IEEE, 2017, PP. 125–130.

[8] J. ROEMER, M. GROMAN, Z. YANG, Y. WANG, C. C. TAN, AND N. MI, Improving Virtual Machine Migration via Deduplication, IN *11th IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS 2014)*. IEEE, 2014, PP. 702–707.

[9] Z. YANG, M. HOSEINZADEH, P. WONG, J. ARTOUX, C. MAYERS, D. T. EVANS, R. T. BOLT, J. BHIMANI, N. MI, AND S. SWANSON, H-NVMe: A Hybrid Framework of NVMe-based Storage System in Cloud Computing Environment, IN *36th IEEE International Performance Computing and Communications Conference (IPCCC)*. IEEE, 2017.

[10] Z. YANG, M. HOSEINZADEH, P. WONG, J. ARTOUX, AND D. EVANS, A Hybrid Framework Design of NVMe-based Storage System in Cloud Computing Storage System, PATENT US62/540 555, 2017.

[11] J. BHIMANI, N. MI, M. LEESER, AND Z. YANG, FiM: Performance Prediction Model for Parallel Computation in Iterative Data Processing Applications, IN *10th IEEE International Conference on Cloud Computing (CLOUD)*. IEEE, 2017.

[12] J. BHIMANI, Z. YANG, M. LEESER, AND N. MI, Accelerating Big Data Applications Using Lightweight Virtualization Framework on Enterprise Cloud, IN *21st IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2017.

[13] T. Hatanaka, R. Yajima, T. Horiuchi, S. Wang, X. Zhang, M. Takahashi, S. Sakai, and K. Takeuchi, Ferroelectric (fe)-nand flash memory with non-volatile page buffer for data center application enterprise solid-state drives (ssd), in *2009 Symposium on VLSI Circuits*. IEEE, 2009, pp. 78–79.
[14] A. Gawanmeh, Optimizing lifetime of homogeneous wireless sensor networks for vehicular monitoring, in *Connected Vehicles and Expo (ICCVE), 2014 International Conference on*. IEEE, 2014, pp. 980–985.
[15] Z. Yang, J. Wang, D. Evans, and N. Mi, AutoReplica: Automatic Data Replica Manager in Distributed Caching and Data Processing Systems, in *1st International workshop on Communication, Computing, and Networking in Cyber Physical Systems (CCNCPS)*. IEEE, 2016.
[16] Z. Yang, J. Wang, and D. Evans, Automatic Data Replica Manager in Distributed Caching and Data Processing Systems, Patent US15/408 328, 2017.
[17] vSphere Hypervisor, www.vmware.com/products/vsphere-hypervisor.html.
[18] J. Tai, D. Liu, Z. Yang, X. Zhu, J. Lo, and N. Mi, Improving Flash Resource Utilization at Minimal Management Cost in Virtualized Flash-based Storage Systems, *Cloud Computing, IEEE Transactions on*, no. 99, p. 1, 2015.
[19] T. Wang, J. Wang, N. Nguyen, Z. Yang, N. Mi, and B. Sheng, EA2S2: An Efficient Application-Aware Storage System for Big Data Processing in Heterogeneous Clusters, in *26th International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2017.
[20] Z. Yang, J. Wang, and D. Evans, A Duplicate In-memory Shared-intermediate Data Detection and Reuse Module in Spark Framework, Patent US15/404 100, 2017.
[21] J. Wang, Z. Yang, and D. Evans, Efficient Data Caching Management in Scalable Multi-stage Data Processing Systems, Patent US15/423 384, 2017.
[22] Z. Yang, J. Tai, J. Bhimani, J. Wang, N. Mi, and B. Sheng, GREM: Dynamic SSD Resource Allocation In Virtualized Storage Systems With Heterogeneous IO Workloads, in *35th IEEE International Performance Computing and Communications Conference*. IEEE, 2016.
[23] Understanding Penalty of Utilizing RAID, theithollow.com/2012/03/21/understanding-raid-penalty/.
[24] Z. Yang and M. Awasthi, I/O Workload Scheduling Manager for RAID/non-RAID Flash Based Storage Systems for TCO and WAF Optimizations, Patent US15/396 186, 2017.
[25] dstat, https://dag.wiee.rs/home-made/dstat.
[26] iostat, https://linux.die.net/man/1/iostat.
[27] blktrace, https://linux.die.net/man/8/blktrace.
[28] K. Chodorow, *MongoDB: the definitive guide*. O'Reilly Media, Inc., 2013.
[29] J. Shi, J. Wan, H. Yan, and H. Suo, A survey of cyber-physical systems, in *Wireless Communications and Signal Processing (WCSP), 2011 International Conference on*. IEEE, 2011, pp. 1–6.
[30] K.-D. Kang and S. H. Son, Real-time data services for cyber physical systems, in *Distributed Computing Systems Workshops, 2008. ICDCS'08. 28th International Conference on*. IEEE, 2008, pp. 483–488.
[31] L. Li, J. C. Snyder, I. M. Pelaschier, J. Huang, U.-N. Niranjan, P. Duncan, M. Rupp, K.-R. Müller, and K. Burke, Understanding machine-learned density functionals, *International Journal of Quantum Chemistry*, vol. 116, no. 11, pp. 819–833, 2016.
[32] M. Wojnowicz, D. Nguyen, L. Li, and X. Zhao, Lazy stochastic principal component analysis, in *IEEE International Conference on Data Mining Workshop*, 2017.
[33] L. Li, T. E. Baker, S. R. White, and K. Burke, Pure density functional for strong correlations and the thermodynamic limit from machine learning, *Phys. Rev. B*, vol. 94, no. 24, p. 245129, 2016.
[34] J. Wang, T. Wang, Z. Yang, N. Mi, and S. Bo, eSplash: Efficient Speculation in Large Scale Heterogeneous Computing Systems, in *35th IEEE International Performance Computing and Communications Conference*. IEEE, 2016.
[35] J. Wang, T. Wang, Z. Yang, Y. Mao, N. Mi, and B. Sheng, SEINA: A Stealthy and Effective Internal Attack in Hadoop Systems, in *International Conference on Computing, Networking and Communications (ICNC 2017)*. IEEE, 2017.
[36] H. Gao, Z. Yang, J. Bhimani, T. Wang, J. Wang, B. Sheng, and N. Mi, AutoPath: Harnessing Parallel Execution Paths for Efficient Resource Allocation in Multi-Stage Big Data Frameworks, in *26th International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2017.
[37] I. Iliadis, E. K. Kolodner, D. Sotnikov, P. K. Ta-Shma, and V. Venkatesan, Enhancing reliability of a storage system by strategic replica placement and migration, Apr. 25 2017, uS Patent 9,635,109.
[38] L. Cheng, S. Kotoulas, T. E. Ward, and G. Theodoropoulos, Robust and skew-resistant parallel joins in shared-nothing systems, in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. ACM, 2014, pp. 1399–1408.
[39] T. Harter, D. Borthakur, S. Dong, A. Aiyer, L. Tang, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, Analysis of HDFS under HBase: A Facebook messages case study, in *Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST 14)*, 2014, pp. 199–212.
[40] A. Cidon, S. Rumble, R. Stutsman, S. Katti, J. Ousterhout, and M. Rosenblum, Copysets: Reducing the frequency of data loss in cloud storage, in *Presented as part of the 2013 USENIX Annual Technical Conference (USENIX ATC 13)*, 2013, pp. 37–48.
[41] J. Zhang, G. Wu, X. Hu, and X. Wu, A distributed cache for hadoop distributed file system in real-time cloud services, in *2012 ACM/IEEE 13th International Conference on Grid Computing*. IEEE, 2012, pp. 12–21.
[42] Z. Yang, M. Awasthi, M. Ghosh, and N. Mi, A Fresh Perspective on Total Cost of Ownership Models for Flash Storage in Datacenters, in *2016 IEEE 8th International Conference on Cloud Computing Technology and Science*. IEEE, 2016.
[43] J. Bhimani, J. Yang, Z. Yang, N. Mi, N. K. Giri, R. Pandurangan, C. Choi, and V. Balakrishnan, Enhancing SSDs with Multi-Stream: What? Why? How?, in *36th IEEE International Performance Computing and Communications*

Conference (IPCCC), Poster Paper. IEEE, 2017.
[44] Z. Yang, M. Ghosh, M. Awasthi, and V. Balakrishnan, Online Flash Resource Migration, Allocation, Retire and Replacement Manager Based on a Cost of Ownership Model, Patent US15/094 971, US20 170 046 098A1, 2016.
[45] J. Bhimani, J. Yang, Z. Yang, N. Mi, Q. Xu, M. Awasthi, R. Pandurangan, and V. Balakrishnan, Understanding Performance of I/O Intensive Containerized Applications for NVMe SSDs, in *35th IEEE International Performance Computing and Communications Conference*. IEEE, 2016.
[46] Z. Yang, S. Hassani, and M. Awasthi, Memory Device Having a Translation Layer with Multiple Associative Sectors, Patent US15/093 682, US20 170 242 583A1, 2015.
[47] Z. Yang, M. Ghosh, M. Awasthi, and V. Balakrishnan, Online Flash Resource Allocation Manager Based on TCO Model, Patent US15/092 156, US20 170 046 089A1, 2016.
[48] Z. Yang, J. Wang, and D. Evans, Adaptive Caching Replacement Manager with Dynamic Updating Granulates and Partitions for Shared Flash-Based Storage System, Patent US15/400 835, 2017.
[49] L. Cheng, Y. Wang, Y. Pei, and D. Epema, A coflow-based co-optimization framework for high-performance data analytics, in *Parallel Processing (ICPP), 2017 46th International Conference on*. IEEE, 2017, pp. 392–401.
[50] Y. Zhao, C. Li, L. Li, and P. Zhang, Dynamic replica creation strategy based on file heat and node load in hybrid cloud, in *Advanced Communication Technology (ICACT), 2017 19th International Conference on*. IEEE, 2017, pp. 213–220.
[51] R. K. Grace and R. Manimegalai, Dynamic replica placement and selection strategies in data gridsa comprehensive survey, *Journal of Parallel and Distributed Computing*, vol. 74, no. 2, pp. 2099–2108, 2014.
[52] R. Li, W. Feng, H. Wu, and Q. Huang, A replication strategy for a distributed high-speed caching system based on spatiotemporal access patterns of geospatial data, *Computers, Environment and Urban Systems*, 2014.
[53] P. L. Suresh, M. Canini, S. Schmid, and A. Feldmann, C3: Cutting Tail Latency in Cloud Data Stores via Adaptive Replica Selection, in *NSDI*, 2015, pp. 513–527.
[54] C. Liu, R. Ranjan, C. Yang, X. Zhang, L. Wang, and J. Chen, MUR-DPA: top-down levelled multi-replica merkle hash tree based secure public auditing for dynamic big data storage on cloud, *IEEE Transactions on Computers*, vol. 64, no. 9, pp. 2609–2622, 2015.
[55] J.-Y. Zhao, M. Tang, and R.-F. Tong, Connectivity-based segmentation for gpu-accelerated mesh decompression, *Journal of Computer Science and Technology*, vol. 27, no. 6, p. 1110, 2012.
[56] J. Zhao, M. Tang, and R. Tong, Mesh segmentation for parallel decompression on gpu, *Computational Visual Media*, pp. 83–90, 2012.
[57] M. Tang, J.-Y. Zhao, R.-f. Tong, and D. Manocha, Gpu accelerated convex hull computation, *Computers & Graphics*, vol. 36, no. 5, pp. 498–506, 2012.
[58] W. Cai, X. Zhou, and X. Cui, Optimization of a gpu implementation of multi-dimensional rf pulse design algorithm, in *Bioinformatics and Biomedical Engineering,(iCBBE) 2011 5th International Conference on*. IEEE, 2011, pp. 1–4.

Edited by: Amjad Gawanmeh
Received: Jun 1, 2017
Accepted: Oct 27, 2017

DOI 10.12694/scpe.v18i4.1331
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 18, Number 4, pp. 313–330. http://www.scpe.org

TESTING AND VALIDATION OF MODBUS/TCP PROTOCOL FOR SECURE SCADA COMMUNICATION IN CPS USING FORMAL METHODS

IRFAN A. SIDDAVATAM*, SACHIN PAREKH, TANAY SHAH, AND FARUK KAZI

Abstract. Cyber-Physical Systems (CPS's) evident representation is Supervisory Control, and Data Acquisition(SCADA). As SCADA is being refurbished with advanced computing and communication technologies, the risk involved in adopting/updating to new technology needs to be validated and verified thoroughly. One of the greatest challenges is security testing of protocols. All CPS systems being live and attached to physical process can not be scheduled for penetration testing and verification. This paper presents design and implementation of industrial compliant SCADA test bed, the formal analysis of semantics and security of Modbus/TCP protocol using Coloured Petri Nets(CPN) tool. A novel method is proposed to differentiate attack vector by identifying influential nodes using formal concept analysis. Modbus/TCP conceptualized attack from analysis is tested and verified on the test bed.

Key words: Coloured petri net, Cyber-physical system, Formal analysis, Modbus/TCP, Communication protocol, SCADA.

AMS subject classifications. 68M12, 14D15, 14D15

1. Introduction. Cyber-Physical Systems(CPS) are highly integrated engineering systems, with a paramount requirement of cohesion and coupling between its computing, communication and control technologies. They have to deal with the physical system with the stringent basis of robustness, efficiency, stability and reliability [1, 2]. CPS have representation in a broad range of areas, like a small scale example of medical devices [3] to large scale CPS of a power grid. CPS perspective of the power grid has received a lot of attention in the research community, as the cyber system embedded in the communication network, substation automation, and control center takes the responsibility to provide safe, secure and efficient transmission, distribution and reliable generation [4]. The smart grid is the next generation of a power grid that will have all IT communication architecture from substation automation along with SCADA to home area network based on Ethernet communication protocol[5].

In our current time, pressing challenges to handle CPS developed on legacy control systems are its vulnerability and patch management. SCADA being an integral part of CPS system, it's vulnerability risk while connected to the public network like the Internet have been known [6]-[16]. SCADA system if compromised will have the severe impact on the power system. It is possible to formulate an attack that can launch vicious switching actions causing disruption in load. Such scenario may be more grievous if the attacker can penetrate the control center of the SCADA system [17]. Till date keeping these machines functional was the prime focus but now making it secure is the biggest challenge. According to SANS Institute survey, 70% of SCADA system operators weigh the risks to be high to severe, and 33% speculate that they may have had incidents [18]. Numerous reports reveal SCADA systems are under increasing attack and are vulnerable. The widely known incidences are Night Dragon Remote Access Trojans (RATs) infected Exxon, Shell, BP, and others for data stealing/spying, Stuxnet infected Iran's Natanz nuclear facility and disrupted nuclear centrifuges, Ukrainian power companies experienced attacks on distribution substation affecting 225,000 customers [19]-[24]. Further, the security breach is exploitation of flaws existing in a system or due to inappropriate update carried out. The basic thumb rule for up-gradation is to 'risk implementing an upgrade should be less than not implementing'. Main contribution of this paper are:

Contribution 1: In CPS, system development or up-gradation is the critical task. As Modbus/Remote Terminal Unit(Modbus/RTU) runs on serial communication, it is considered to be secured but if it is upgraded to Modbus/Transmission Control Protocol(Modbus/TCP) will turn the system vulnerable to cyber-attack. To understand the impact, we contribute by providing formal specification model of Modbus/TCP using CPN and perform its security analysis.

Contribution 2: Our contribution is focused towards verification by the approach of testing assumptions and not by testing everything. Standard assumptions like unreliable communication channel that are

*Electrical Department, Veermata Jijabai Technological Institute, H R Mahajani Marg, Matunga, Mumbai, Maharashtra, India 400031 (irfanasv@gmail.com).

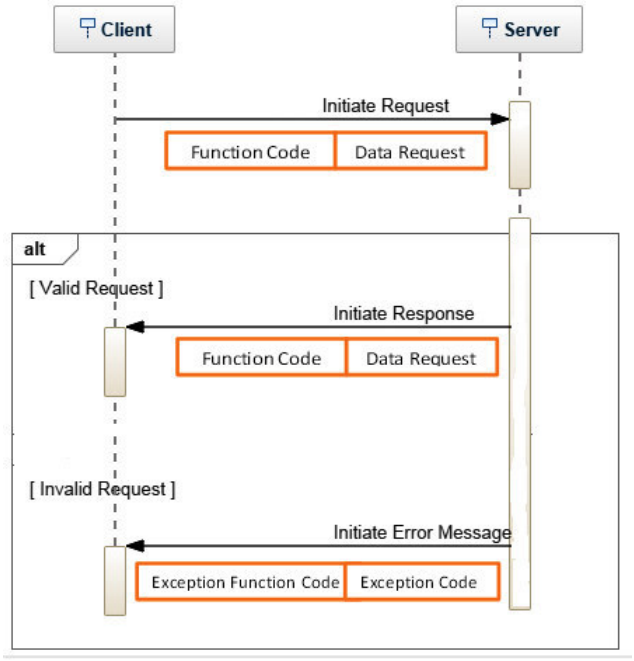


FIG. 2.1. *Client-Server interaction*

required for security assurance of protocol are tested. Our novel method performs state space analysis using CPN which is developed to verify and validate the concept of attack using formal concept analysis.

Contribution 3: With many of the SCADA systems being significantly dated, security was little concern before today's internet age, for this reason, most control systems are open to attack from the outside. We contribute towards mitigation of this risk by the development of SCADA test bed that can be used for security evaluation, testing, and simulations.

Contribution 4: Deceptions attack vector conceptualized from modeling is tested experimentally on research platform; a proposed solution for mitigation of attack on SCADA is demonstrated.

2. Security Analysis of Modbus/TCP Protocol.

2.1. Modbus/TCP Protocol. In 1979, Modicon developed Modbus protocol for communication over single twisted pair cable with many devices. Modbus was originally designed for RS232, and it is also adapted for RS485. Modbus protocol has true multi-drop network along with faster speed over longer distances. It is now royalty-free protocol. Modbus works in master-slave mode, wherein master can communicate with one or many slaves. Master or Client can initiate a transaction (called query) to which slave or server responds by providing requested data. A slave is sensing or measuring device like I/O transducer, valve, network drive, or other measuring instruments which processes information and sends its output to the master using Modbus.

The general Modbus frame has two data units. First, Application Data Unit(ADU) that deals with communication layer and also has additional fields to map protocol on specific bus or network. Second, Protocol Data Unit(PDU) is initiated by the client for the transaction. The query contains a server address, a function code specifying the action to be carried out, data and error checking field. As shown in Figure 2.1, if the server receives PDU appropriate to a request with no error, it responds back confirming action performed, data and error checking field. In case if an error is detected in operation, the server responds by sending exception function code and exception code. Modbus/TCP is simply the Modbus/RTU protocol with a Transmission Control Protocol(TCP) interface that runs on ethernet.

Figure 2.2 shows the Modbus frame. Following is the description of each field:

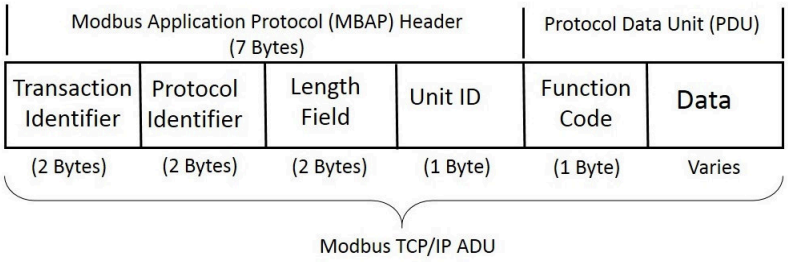


FIG. 2.2. *Modbus/TCP frame format*

Transaction/invocation Identifies (2 Bytes): On same TCP connection, if the client wants to send multiple messages without waiting for responses, transaction identifier field is used for pairing such request.

Protocol Identifier (2 Bytes): This field is set to '0' for Modbus services, for the future extension all other are kept reserved.

Length (2 Bytes): Length field contains byte count of all the remaining fields including the data field, function code, and unit identifier byte.

Unit Identifier (1 Byte): If a remote server is located on non-TCP/IP network, unit identifier field identifies it. In the case of Modbus TCP/IP server application, the unit Identifier is set to '00 or FF'.

Function Code (1 Byte): The function code is an indication of the type of action server has to perform. The function code field of a Modbus/TCP data unit has the length of one byte. The correct hexadecimal function codes are in the range of 1 - 127, while remaining are kept reserved and used as exception responses (128 - 255 Hex). Function code cannot have value '0' [25, 26].

2.2. Formal Modelling of Protocol.

2.2.1. Description of Protocol. As explained in the previous section, Modbus/TCP communication involves two entities, i.e., client and server. The client is the initiator of communication and displays result obtained from the server. Modbus/TCP protocol fields as Transaction identifier, Protocol identifier, Length, Unit-identifier facilitates the establishment of a connection between client and server. These fields remain agnostic to a behavior of process hence are ignored in modeling semantic.

Modbus/TCP protocol consists of three types PDU; Request PDU is initiated by a client, and it includes parameter function code and request data. The Response PDU is generated by a server having function code and data, and Exception PDU contains exception code in case of error. Accordingly, Modbus/TCP client process can be formally described as:

Client Process: A Client process equipped with function code f_c and data d_a and using server device 'b' is:

$$
\begin{aligned}
&Client(a, f_c, d_a) = \\
&- \qquad env?b : Device \rightarrow send.a.b\langle f_c, d_c\rangle \\
&- \qquad \begin{array}{r} \Box \\ f_c \in F \\ d_c \in D \\ e_c \in E \end{array} \left(\begin{array}{c} recives.b.a\,\{f_c, d_c\} \rightarrow STOP \\ \\ recives.b.a\,\{f_c, d_c, e_c\} \rightarrow STOP \\ \\ \end{array} \right)
\end{aligned}
$$

where F and D are the set of all objects that client node can accept as function code and data, used to query server 'b', while E is the set object of exception from which client can receive in case of error. The initial communication 'env?b:Device' is a representation of how depending upon physical process's might use any device to open a session with agent b [27]. Client process has two choices of event one that it receives a response to successful execution of 'f_c' , and other is the exception raised due to error.

The transition rules for external choice '$\Box$' reflect the fact that the first external event resolves the choice in favor of the process performing the event.

Server Process: The server role has a program that validates the 'f_c' and 'd_c' value received, successful execution of a function and then responds accordingly. we get the following descriptions:

$$Server(b, f_c, d_a) =$$

$$\begin{array}{r} \Box \\ f_c \in F \\ d_c \in D \\ e_c \in E \end{array} \left(\begin{array}{r} recives.a.b\,\{f_c, d_c\} \rightarrow \\ \\ | \; notValid.f_c \rightarrow send.b.a\,\{f_c, d_c, e_c\} \rightarrow STOP \\ | \; notValid.f_c.dataAddress \rightarrow send.b.a\,\{f_c, d_c, e_c\} \rightarrow STOP \\ | \; notValid.d_c \rightarrow send.b.a\,\{f_c, d_c, e_c\} \rightarrow STOP \\ | \; notExecuted.f_c \rightarrow send.b.a\,\{f_c, d_c\} \rightarrow STOP \\ | \; send.b.a\,\{f_c, d_c, e_c\} \rightarrow STOP \end{array} \right)$$

Obviously, in this case, it is not the server process's instigates the protocol, but rather it is started when it receives a message from an initiator. The server executes function code validates each parameter and its possible execution on a device if any error occurs appropriate exception is raised and send to the client. In case of proper request and execution of a command, the response contains values of 'f_c' and 'd_c' executed on the devices.

Attacker Process: The security concerns of the protocol are modeled by adding an attacker or intruder process into the network. Further, through out the literature term, attacker and intruder are interchangeable. The attacker can tamper with the messages that pass around. Intruder process has a knowledge K that contains information such as members involved in the operational network, protocol details, etc. It can listen to message flow all the time. The attacker can formulate messages inferring K that it has gathered over a period and the messages received till that moment.

$$\begin{array}{ll} Attacker(X) = learn?c : commands \rightarrow Attacker(deduce(X \cup \{c\})) \\ - \qquad\qquad \Box say?m : X \cap messages \rightarrow Attacker(X) \end{array}$$

Here, the parameter X ranges over function code 'f_c' and data 'd_c' all PDU that attacker can generate. '*commands*' is the subset of fact that represent all PDU that might be generated or accepted over the communication medium by process. The function deduce(X) calculates all PDU developed from X.

Network Model: The network consist of three entities Client, Server and Attacker.There are different models of interconnection used to put together these entities [28]. We have modelled the intruder as being the communication medium. This network is connected together via renaming that match up the *receive:a:b: m receive:a:b:m* and *send:a:b:m* events performed by agent and server processes with the *say:m* and *learn:m* events of the intruder. The *send* and *learn* channels are connected by renaming them both to a *take* channel, and the receive and say channels are connected by renaming them to a *fake* channel. Network combined together is described below:

$$\begin{array}{ll} Client[[fake, take/receive, send]] \\ ||| \; Server[[fake, take/receive, send]] \\ - \qquad\quad \| \; Attacker[[take.x.y, fake.x.y/learn, say \mid x, y \in \{Client\} \cup \{Server\}]] \end{array}$$

Figure 2.3 shows this network, the dotted lines represent attacker channel inclusion by renaming, to keep thing simple only a subset of the channel names are included in the diagram.

2.3. Modbus/TCP Model in CPN. Petri net creates graphical models of process, formalization of these net is based on rigorous mathematical semantics. Petri net models can be used to describe and study many information processing systems that are characterized by being distributed, parallel, asynchronous, concurrent and non-deterministic. CPN [29] is the methodology used for modeling and verification of concurrent systems.

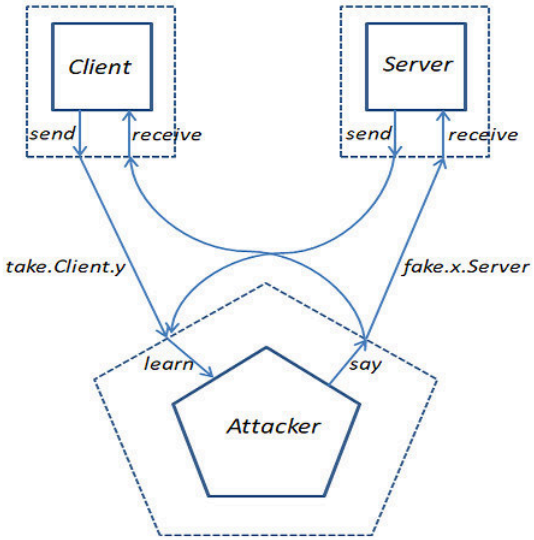


FIG. 2.3. *Network with attacker included*

CPN [30], as a derivative combines these capabilities of Petri nets with a high-level programming language, Standard Meta Language(SML) [31, 32] to provide the primitives for defining data types, data manipulations and creating compact graphical models.

2.3.1. CPN Block Diagram for Modbus/TCP Scenario. Modbus/TCP is complete protocol suite that caters to all range of industrial device as RTU, Intelligent Electronic Device(IED), Variable Frequency Drives(VFD) and many other. Depending upon the nature of the application, client issues function code as the request that set or read required data model. As a test case, we model scenario that writes single register. A most widely used industrial application for this situation is, VFD set from the operator workstation to run an induction motor at the required speed. Figure 2.4(a) show typical set up for mentioned scenario and Figure 2.4(b) shows block diagram for CPN model. The requests are formed with packets at client then transmitted through the network. As the request passes through the network it can be tampered by the attacker or can just be passed on depending upon the criticality of the command. As the server receives the command, it is validated then passed on for processing. In the case of error, the exception is raised and the response is formulated with the error code. If the request is properly processed, response consists of function code and the current value of the device after execution of the command. The function codes and other parameters of Modbus/TCP are used for modelling with data, functions, and headers to mimic actual behaviour of the protocol.

2.3.2. CPN Tool Initial Model . An initial model developed is to validate and analyse general behaviour of Modbus/TCP protocol under no threat condition. The model describes normal behaviour with all error condition being simulated. Figure 2.5 is complete model developed in CPN Tools version 4.0.1.

A. Declaration: The important color set is the one that describes request PDU. The request PDU has been defined as a product of three basic color set: First one is *Function* it represents the function code required to write single register. Since function code is designated using hexadecimal notation, an example of valid function code description is '*0x0006*'. The second color set is *Register Address* declared as '*RAdress*' represent valid register address, to keep the model simple values are restricted to two values '*0x001*' and '*0x002*', rest all are considered illegal. The third one is *Register Value* declared as '*RValue*' represents register value or data. The register value depends upon the process to model constraint involved in the process. We have restricted value of the register to in range $10 \leq RValue \leq 45$. In case of response, if the operation result to erroneous along with the all color set of request PDU color set *exceptioncode* is also added. Color set *exceptioncode* is of string type and is set with exception code depending upon the error occurred. As shown in Figure 2.6 there are three functions declared, function *correctValue* evaluates the valid range of register data, *correctAdd* inspects

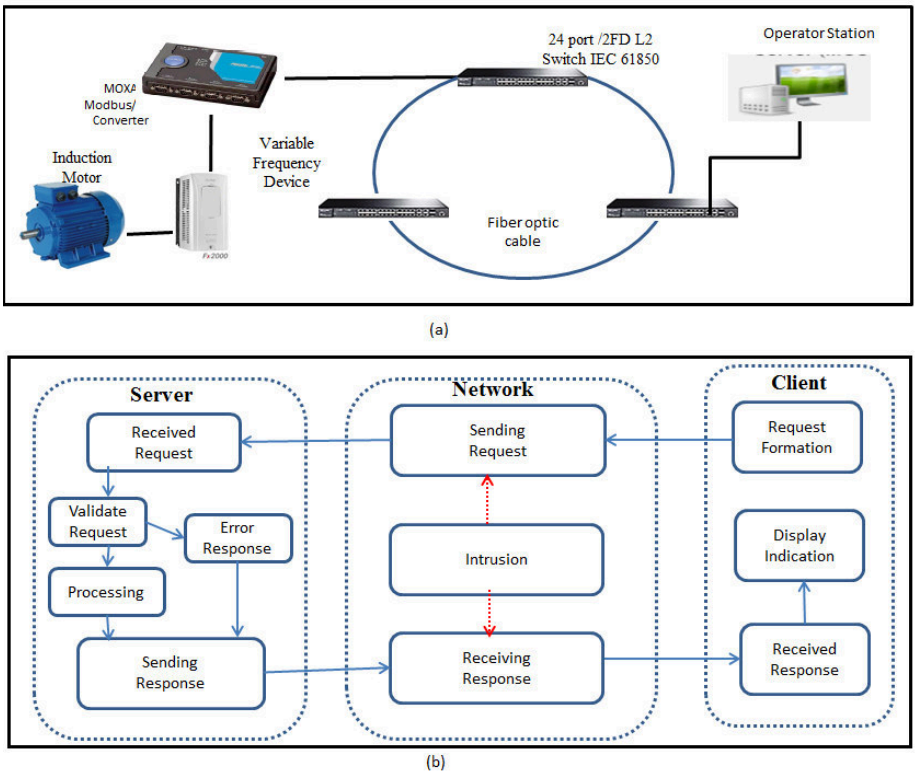


FIG. 2.4. *(a) Implementation scenario (b) CPN Block diagram*

register address assigned. Function *errorcode* figures out if the exception is raised.

B. Operations: Figure 2.5 depicts initial model for Modbus/TCP protocol for the scenario wherein client executes the command that writes single register of the severing device. In this figure, the client station composes a request and transmits the request through the network to the server. The initial composition of a request is through following Datasource or Place: '*Funcode*', '*RAddress*' and '*RegValue*'. These places have markings that form the initial state of CPN model. The occurrences of the markings create a token on Place '*Request*' . '*Request*' is assigned with a color set '*Request*'. Colour set '*Request*'. is a product of the data type '*fcode*', '*RAdd*' and '*RegValue*'. Similarly, when the incoming arc of the transition Sending Request is satisfied, tokens '*fcode, RAdd, RegValue*' is created representing request packet formation. The major transitions that need to satisfied are:

SENDING REQUEST: The transition has Boolean flag 'fstat' to be satisfied to place token further. The flag represents a satisfied establishment of a connection by the TCP protocol.

RECEIVED REQUEST: It is one of the important transaction; it validates token for correct function code, register address, and register value. The evaluation is carried out using the function declared as shown in Figure 2.6, if Request with error is encountered, error token is passed.

REQUEST PROCESSING: This transition models the behavior where in error token is raised if the processing of valid token fails.

ERROR RESPONSE: Transition captures the type of error occurred and passes on exception code raised.

RESPONSE FORMATION: In Modbus/TCP, the response is generated in both cases for normal execution and even if the exception is raised. Transition fires according to token arrived to form appropriate response token.

2.3.3. CPN Tool Attack Model. In order to perform security assessment of protocol, Initial model is further extended by adding Intruder as communication medium. Intruder or the attacker is characterised as

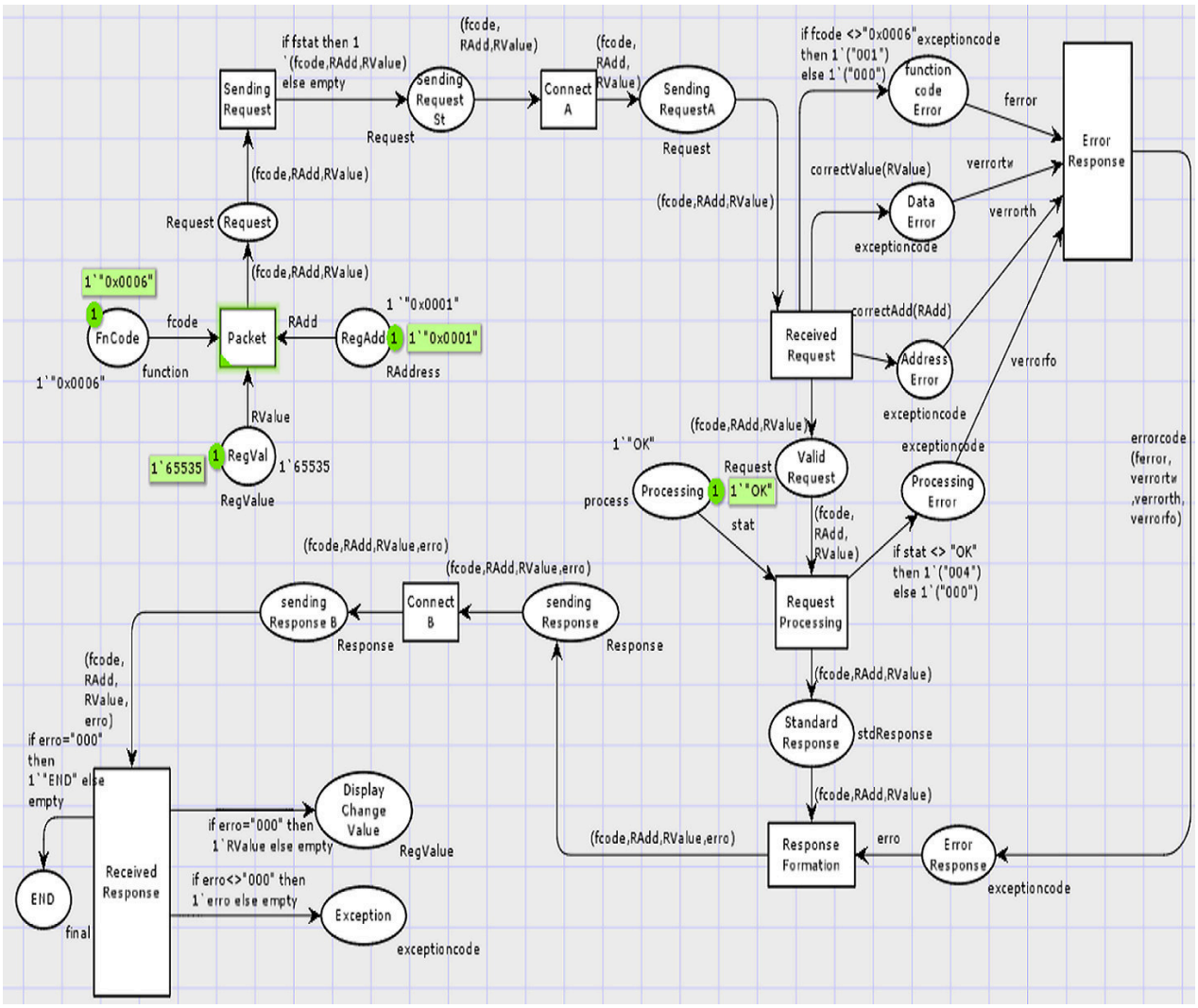


Fig. 2.5. *CPN initial model*

explained in *Sect. 2.2.1.* Figure 2.7 elaborates in detail the attack model.

A. Declaration: Attack Model includes Place '*Attack Enable*', this Place decides the mode of communication channel. Mode of communication channel can be changed by setting '*value*' token of Place '*Attack Enable*'. Channel can be configured into following ways. 1. 'General Mode' by keeping the token 'empty', in this mode the model behaves as 'Initial Model'. Communication channel works as reliable and secure medium, having no attack generated. 2. If string '*insecure*' is set as the token, channel turns to be insecure and false data injection attack is executed. The attack changes variable '*RValue*' to predefined value intend to cause damage in physical system. 3. In this mode Denial of Service(DoS) attack is executed, by setting token '*add*' or '*code*'. Here set token changes the value of '*RAdd*' or '*fcode*' causing a legitimate PDU to converted into erroneous PDU intern avoiding the server to respond in the required manner.

B. Operations: Figure 2.7 describes complete details of Attack Model. Following are the transitions that play a significant role in the operation of model:

Attack Up Stream: To carry out false data injection or DoS attack, in its first phase client request needs to tamper while it is routed to the server. Transition '*Attack Up Stream*' fires as per the type of token provided to change the parameters that are sent towards the server. This is the first phase wherein

CPN Tools (Version 4.0.1, February 2015)

```
Declarations
  Modbus declarations
    var ferror,verrortw,verrorth,verrorfo:exceptioncode;
    fun correctAdd(RAdd:RAddress)
      =if RAdd="0x0001" then 1`("000")
      else
      if RAdd="0x0002" then 1`("000")
      else 1`("002")
    fun correctValue(RValue:RegValue)
      =if RValue < 10 then 1`("003")
      else
      if RValue > 45 then 1`("003")
      else 1`("000");
    fun errorcode(ferror:exceptioncode,
      verrortw:exceptioncode, verrorth:exceptioncode,
      verrorfo:exceptioncode)
      = if ferror<>"000" then 1`("001")
        else if verrortw<>"000" then 1`("002")
             else if verrorth<>"000" then 1`("003")
                  else if verrorfo<>"000" then  1`("004")
                       else 1`("000");
```

FIG. 2.6. *Declared function in initial model*

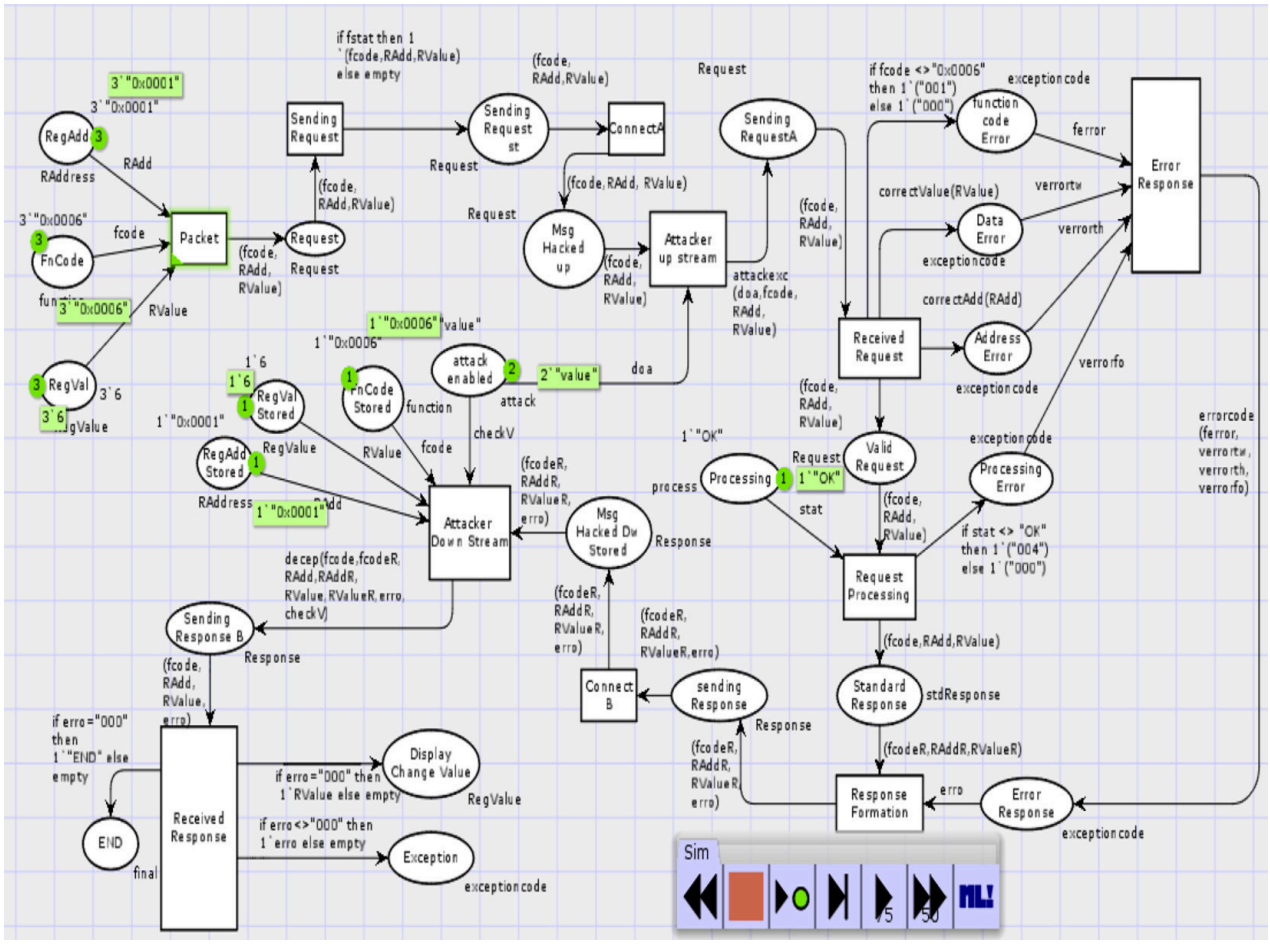


FIG. 2.7. *CPN tool attack model*

TABLE 2.1
Test case for Verification

Test Case	Description	Function Code	Reg. Value	Reg. Address	Status	Attack Flag
1	Normal input with all correct value	'0x0006'	20	'0x0001'	Ok	No Attack
2	Normal input with all correct value error while processing	'0x0006'	20	'0x0001'	Error	No Attack
3	Error in Function code	'0x0009'	20	'0x0001'	Ok	No Attack
4	Error in Register Address	'0x0006'	20	'0x0005'	Ok	No Attack
5	Error in Register Value	'0x0006'	50	'0x0001'	Ok	No Attack
6	Change in Function code by attacker	'0x0009'	20	"0x0001'	Ok	Attack
7	Change in Register Address by attacker	'0x0006'	20	'0x0005'	Ok	Attack
8	Change in Register Value by attacker	'0x0006'	50	"0x0001'	Ok	Attack

request PDU is defaced.

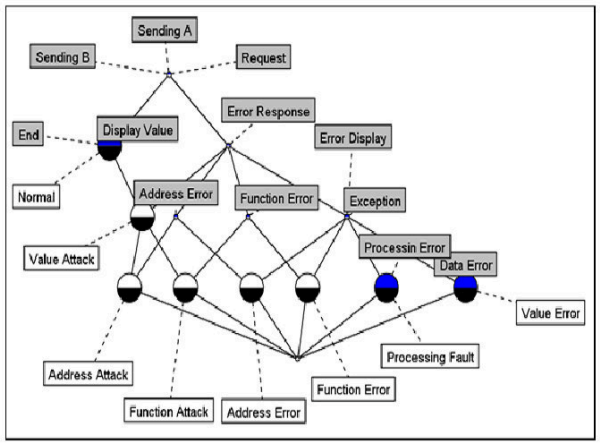
ATTACK DOWN STREAM: In the second phase, a response from the server regarding a status of the request needs to be altered. Forementioned is needed to hide attack execution from the client. Transition '*Attack Down stream*' replaces tampered request with the original one.

2.3.4. Model Validation and Verification. The aim of this section is to validate and verify the correctness of the Modbus/TCP protocol and the attack model. We carry out a different set of the protocol run with the various set of token that simulates the normal behavior and behavior under attack. Multi-set tokens could have been used, but to keep understanding clean and simple single set is used to formulate one PDU transaction. Table 2.1 shows different combination of inputs and expected activation of the features. At this point, data generated by state space analysis need to be analyzed. We choose Formal Concept Analysis (FCA) widely used by information scientists as it performs data analysis, information management, and knowledge representation [33] [35]. State space report generated for the test case indicates the flow of token at each place. As per the concept of FCA, we consider test case example '*function attack*' as the '*formal object*' and Places that carry token during simulation as the '*formal attribute*'. A combined set of the object in relation with the attribute is called as '*formal context*' shown in Figure 2.8.a. One of the primary benefits of FCA is we can visualize the context in line diagram called as '*concept lattice*'. The context for our test case shown in Figure 2.8.a can be represented by concept lattice shown in Figure 2.8.b. Concept lattice is the formation of set concepts of a formal context and the sub-concept and super-concepts relation between the concepts. As shown in figure node represents the concept, in our case for example normal operation represented by node '*normal*' representing concept; similarly, we have concepts '*value attack*', '*error value*', '*function attack*' so on and so forth. We can further drill down to understand the associated concept called as subconcept and the attribute.

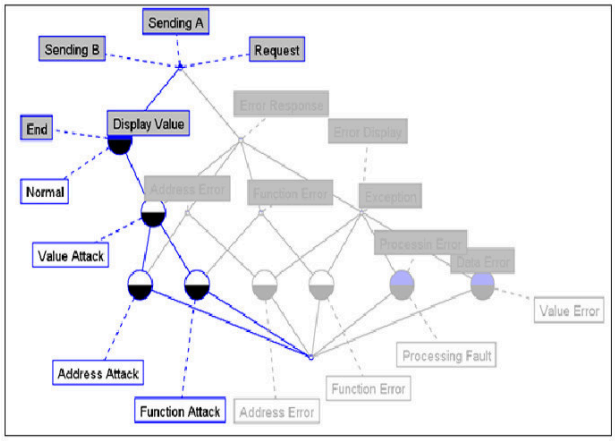
Further analysis is carried out by selecting each node, this process leads us to a conclusion that all attack node, i.e., '*value attack*', '*function attack*', and '*address attack*' are subconcept of concept'*normal* '. Further, concept 'error value'does not have any associated concept. Comparing the attribute of concept '*normal* ' and subconcept '* *attack*' which represent sub-concept of attack, we derive influential Places that are '*END*' and '*Error Response*'. Experimentation of FCA is carried out using 'Concept Explorer'(ConExp) tool [36]. In the following discussion, we analyze the attributes to detect pattern for normal operation, and the deception attacked.

	Request	Sending A	Function E...	Data Error	Address Error	Processin Error	Error Response	Sending B	Display Value	Exception	Error Display	End
Normal	X	X						X	X			X
Processing Fault	X	X				X	X	X		X	X	
Function Error	X	X	X				X	X		X	X	
Value Error	X	X		X			X	X		X	X	
Address Error	X	X			X		X	X		X	X	
Value Attack	X	X					X	X	X			X
Function Attack	X	X	X				X	X	X			X
Address Attack	X	X			X		X	X	X			X

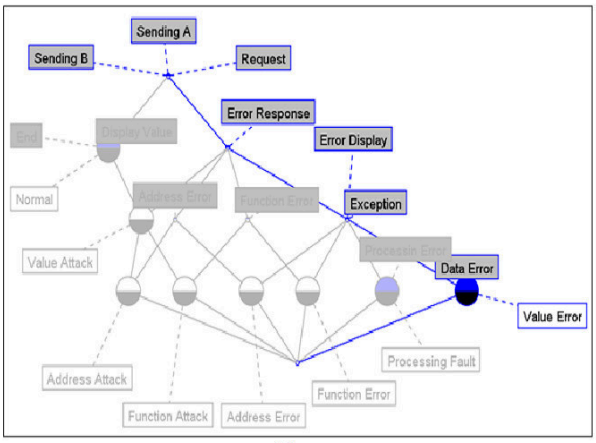
(a)



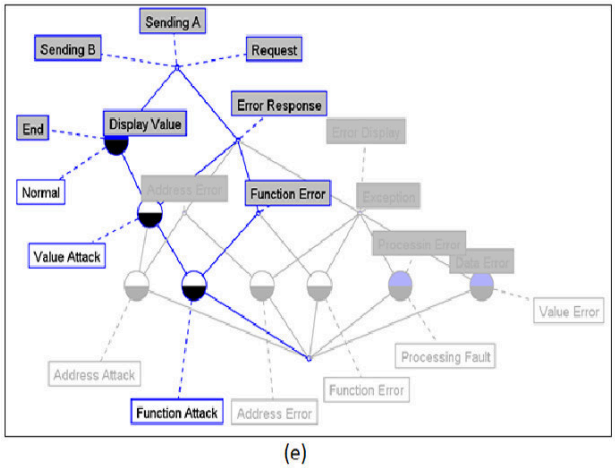
(b)



(c)



(d)



(e)

FIG. 2.8. *(a) Formal context (b) A concept lattice for the formal context in figure a (c) A subconcept-superconcept relation for 'Normal ' node (d) A subconcept-superconcept relation for 'Error value ' node (e) A subconcept-superconcept relation for 'Function attack ' node*

Table 2.2 shows the state space for the defined test case. Please note that '*empty*' value of '*End*' Place is indicated by '10' and if it has token 'END' its value set as '20'. Similarly, '*Error*' place with error token as '15' and without error as '30'. Figure 2.9 shows radar plot of different test case parameters. Executed deception attack follows the same trace as of normal operation. Proper termination is achieved with Place '*END*' acquiring the valid token, even though the value of PDU is changed. To detect attack, if we verify '*error*' Place points out the discrepancy. It is expected to have error value to be '30' if END token is reached, but in all cases, it has '15' that gives us a pattern of execution of an attack.

TABLE 2.2
State space for Results

Test Case	1	2	3	4	5	6	7	8
Nodes	13	13	13	13	13	13	13	13
Arcs	12	12	12	12	12	12	12	12
Dead	4	4	4	4	4	4	4	4
End	20	10	10	10	10	20	20	20
Error	30	15	15	15	15	15	15	15

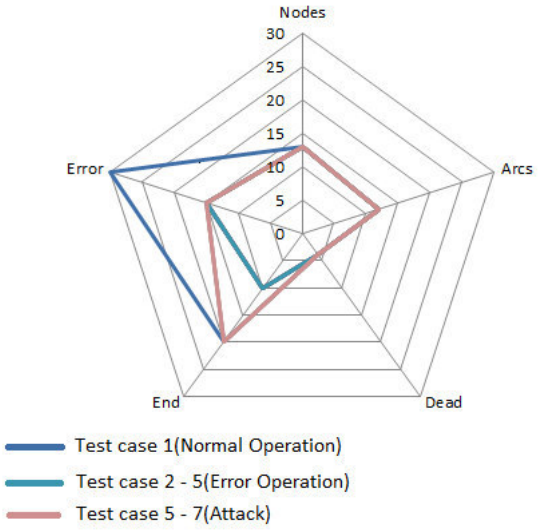


FIG. 2.9. *Graph showing attack pattern*

3. CPS Test bench. As described by Rajkumar et al. [1] CPSs is physical and engineered systems, whose operations are monitored, coordinated, controlled, and integrated by a computing and communicating core. There is definite requirement in research community for having Cyber-Physical Testbed, that aims towards following objective:

1. Design and implementation a SCADA bench for security evaluation, testing, and simulations are necessary to guarantee the safety of our critical infrastructure.
2. Comprehend the impact of cyber-attack such as false data injection, unauthorized command execution, denial of service(DoS) attack, etc. and discover variant for CPS.
3. Testbed that integrates industry standard hardware, software, and Wide Area Measurement System (WAMS) components and protocols.
4. Platform that enables vulnerability assessment of WAMS and related devices.

3.1. Survey of Cyber-Physical Test beds. The interoperability of devices and implementation of functions are the key focus during the construction of smart substations, utilities and manufacturers over cyber security issues and testing of same. Nevertheless, researchers around the world are making an effort on developing cyber-physical Testbed that can develop realistic solutions that can act as 'hot fixes' to the cyber security vulnerable industrial environment. In last one-two years, there are a good number of CSP Testbeds developed, few related are discussed further.

The work Ceeman B. Vellaithurai et al. [37] presented has an overview of real-time cyber-physical testbed developed using ns-3 a discrete-event network simulator and Real-time Digital power system Simulator(RTDS). The testbed uses a combination of simulated/emulated, and physical devices, which allows for an easily reconfig-

TABLE 3.1
Comparison of existing CPS Test bed

Paper	Physical Process	Devices	Protocol	Comm. Network	SCADA	Historian	DCS
[38] Jun-15	RTDS-HIL, Opnet Modeler	LabVIEW, PXI Module	Modbus-TCP	Ethernet Switch	No	No	No
[37] Nov-15	RTDS-HIL	SEL 421 protection relay	IEC 61850	Simulated	No	No	No
[39] Oct-15	RTDS-HIL	sync., Meter	IEC 61850	Switched with copper and fiber optic connections	Yes	No	No
[40] May-16	RTDS-HIL	No	OPC	CORE Emulator	No	No	No
Proposed	OPEL-RT	IED, RTU, EM, VFD	Modbus, IEC 61850, DNP3	Switched with copper and fiber optic connections	Yes	Yes	Yes

urable system. Different types of traffic, such as DNP3 and C37.118.1 PMU data that could be passed through the cyber network emulated in ns-3. In work by Bo Chen et al. [38] RTDS is used to carry out power system simulation, while SCADA and IEDs are simulated using LabVIEW and PXI modules. Paper by Y. Yang et al. [39] discussed the development of cyber-physical test bed that examine the IEC 61850 based smart substations for the impact of cyber-attack and potential cyber security vulnerabilities. V. Venkataramanan et al. [40] developed a real-time, cyber-physical co-simulation using RTDS and Common Open Research Emulator(CORE) to simulate power system and emulate communication network respectively. In work by Uttam Adhikari et al. [41], RTDS hardware-in-loop(HIL) is used to simulate physical process for developed WAMS cyber-physical test bed. Table 3.1 compares presented test bench with existing ones.

3.2. System Description. Testbed should include hardware components, control and configuration software packages, communication networks and protocols, and custom software for applying stimulus and capturing data. The developed Testbed uses communication protocols, software, and hardware that conforms to industry standard. Industrial Control System(ICS) is most referred the case of Cyber-physical system; CPS testbed developed tries to cover all aspect of ICS with electrical system as the physical process. Distributed control system(DCS), SCADA and other control systems those usually found in industrial sectors and critical infrastructures like Programmable logic controllers(PLC) fall under the broad category of the ICS. The primary difference between SCADA and DCS or PLC controlled sub-systems is, former are geographically dispersed while later are confined to factory or plant-centric area. Local area network(LAN) technologies used by DCS and PLC are more high speed and reliable in comparison to the long-distance communication systems employed by SCADA [42]. Figure 3.1 shown is schematic of cyber-physical testbed developed in-house. Each of the components of the testbed is described in following sections.

3.2.1. Physical System. A core aspect of any cyber-physical system is its physical process. The developed testbed uses real-time HIL OPEL-RT simulator OP5600-eMegaSim. OP5600 is used to simulate up to 30 bus power system. There is a constraint on the size of the system but modeled system very precisely imitate the behavior of real power system. The simulator supports C37.118, IEC 61850 protocol stack for communication with laboratory devices and monitoring system.

3.2.2. Distributed Control System(DCS). The DCS is a control system which formulates the control logic and makes a decision of action to be performed on the data collected from the field devices. DCS installed is

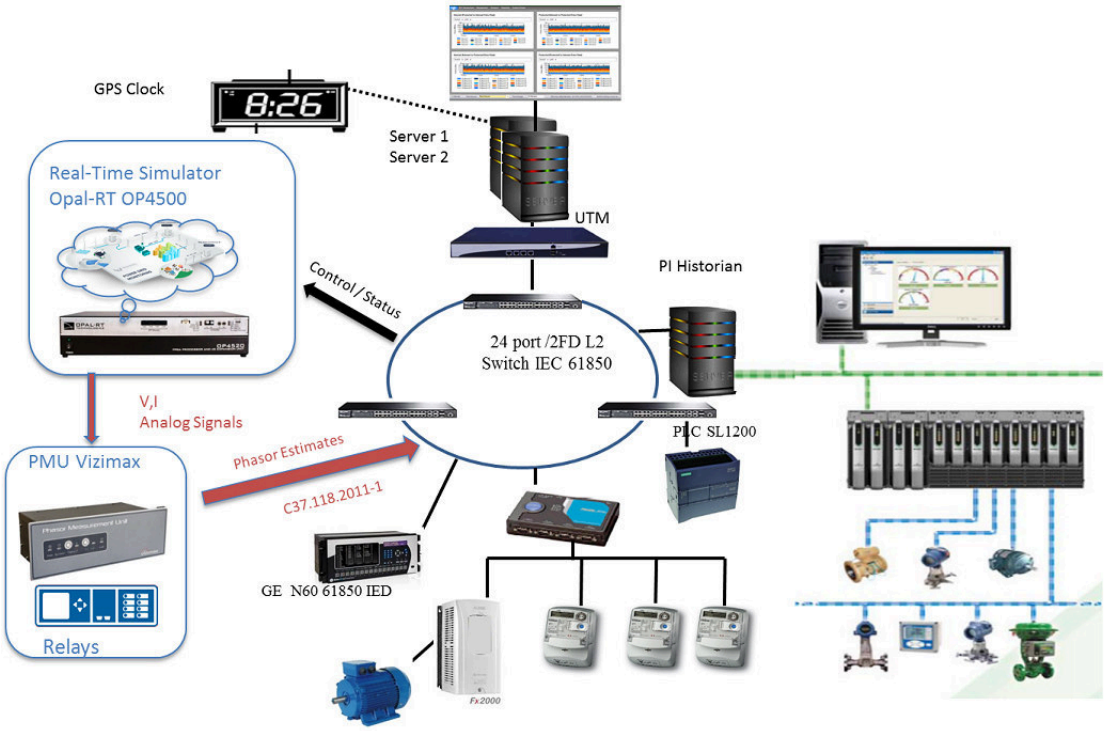


FIG. 3.1. *System architecture of CPS test bed*

of Emerson make. Installed system has a simplex controller, 8 wide I/O interface carrier, Smart wireless gateway 1420 (Copper Ethernet connection, locally mounted antenna), Analog Input/Output, 16 channel, 4 - 20 mA, HART, Discrete Input, 32 channel, 24 V DC, Discrete Output, 8 channel, 24 V DC and DeltaV PredictPro 20 MPC Outputs software. The SCADA Testbed developed has all facets of industrial grade devices. The system supports DNP3, Modbus, IEC 60870-103, 101, and IEC 61850 protocol for communication. GPS clock of SYNTIME make with 10/100 MBPS NTP/SNTP output is used for the time synchronization of the relays and the status points at different locations. A high-speed data is achieved by industry standard networking having an optical fiber backbone operating at 100 Mb per second. The control center was designed to meet the research and training requirement. The iVisionmax is the SCADA software used in the laboratory. The iVisionmax supports major equipment and protocol. Few critical devices are listed below:

Phasor Measurement Unit: PMU measures on the global time reference the magnitude and phase angle of an analog and /or drived phasor, as per the synchrophasor standards (IEEE 1344, IEEE C37.118). Vizimax PMU010000 is installed, it has been tailor made to be compatible with OPL-RT from OEM itself.

OSI-PI: The PI System is historian and also has its ability to collect, analyze, visualize and share large amounts of high-fidelity, time-series data from multiple sources to people and systems across all operations.

Network: As per industrial standard communication channel is of Fiber optic cable(FOC). To have fault tolerant network, ring topology of IEC 61850-3 compatible Hirschmann Layer-2 switches are formed. Segregation of control network and Corporate is achieved with the help of Layer-3 CISCO Switch.

PLC (Siemens S7 1200): Programmable logic controller is used as a local controller. Ladder programming is used with Simatic S7 TIA portal to program a PLC.

Modbus/TCP Gateway: A Modbus/TCP gateway is used as a protocol converter, which encapsulates a Modbus -RTU frame into Modbus/TCP packets to communicate with server and workstation.

Variable Frequency Drive (VFD-Fx-2000 LNT): VFD controls AC motor speed and torque by changing motor input frequency and voltage. Frequency is directly related to the motors speed (RPMs).

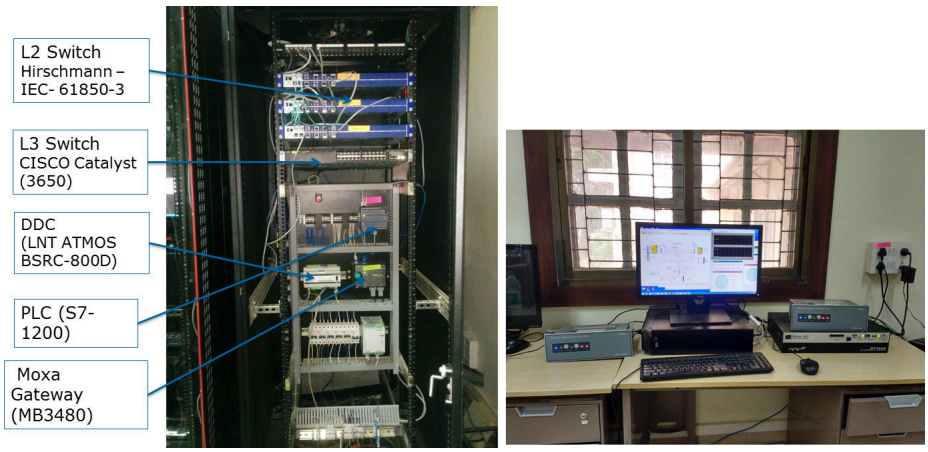


FIG. 3.2. *Test bench*

4. Experimentation and Results.

4.1. Implementation of Formally conceptualised Deception attack on Modbus/TCP. Vulnerability mentioned in *Sect. 2.3* for Modbus/TCP is exploited to form attack vector. This attack execution methodology and its outcomes are explained in following section.

4.1.1. Discovery. Discovery is the initial phase where all information regarding network is gathered. The focus is to know network architecture, communication protocol, etc. Discovery phase is carried out for both phases of attack Upstream and Downstream or deception. Modus operandi for discovery phase is as follows:

$STEP-1:$ Nature of attack considered is of type '*Insider Attack*' wherein the attacker has access to any of the computers on the control network. The attacker has the access physically or gained remote access due to a compermised machine.Attacker maps all devices connected to the network in terms of IP address of the network, port accessed and other network parameters.

$STEP-2:$ In this step, we carry out the Man in the Middle(MITM) attack on each of the connected devices for a fixed time 't'. MITM enables us to sniff the communication between all devices on the network. Increasing the time 't ' helps to get more data to analyze while decreasing 't' decreases the probability of detection.

$STEP-3:$ In the final step, we analyze the sniffed data to identify the network architecture. Fig.4.1(a) shows result of sniffing data on experimental setup shown in Fig.4.1(b). It is quite evident from frequency analysis of communication, and one can detect protocol and interacting devices. For the experimental setup, IP address for SCADA server is 192.168.0.21 and Modbus gateway as 192.168.0.20 communicating over Modbus/TCP. Forementioned communication pattern could be easily detected as histogram shows concentrated Modbus/TCP packets with these IP's.

4.1.2. Deception Attack. Deception attack is a combination of MITM and Denial of Service(DoS) attack. It modifies operations of the field device in such a manner that changes in operations are not reflected at SCADA HMI. The attack may lead to a delayed or no control action being taken by supervisory control, causing failure in the system. The execution steps to formulate attack are as follows:

Step 1: Initial step is to footprint the target network. The result of discovery phase lead us following pointer :

- As there was a convergence of communication at the particular device, indicated network consists of the gateway.
- IP address of network machines were enumerated.
- Frequency of communication suggested the presence of SCADA server and gateway.

Step 2: Inferring from Step 1, the attack will disrupt the original communication path between the server and the gateway. Then we set up a custom path that inserts the local host computer(attacker controlled machine) in between. Forementioned, a man in the middle arragement routes all packet between server

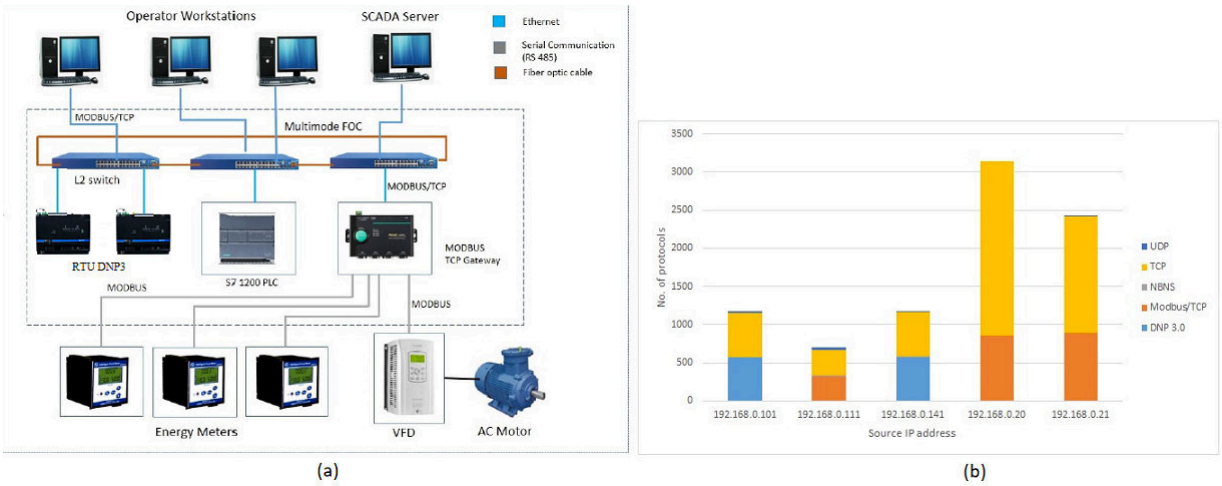


FIG. 4.1. *(a) Experimental set-up for Modbus attack; (b) Packet analysis network*

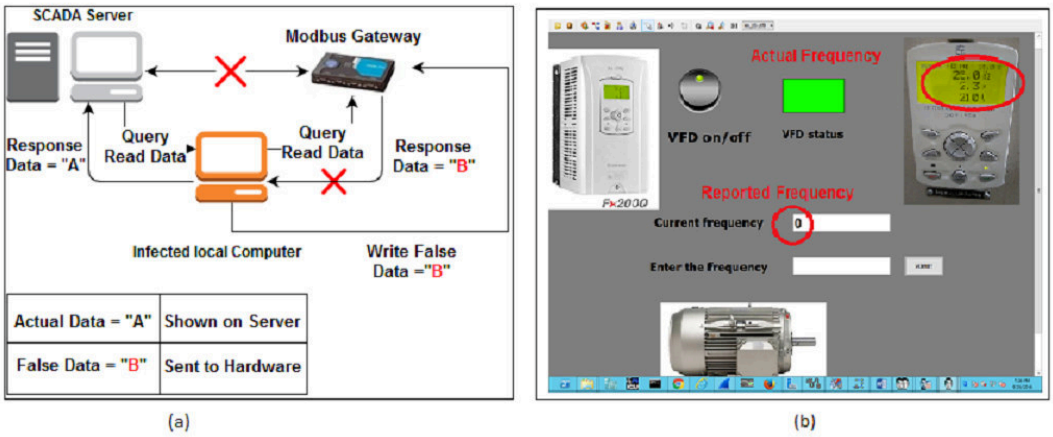


FIG. 4.2. *(a) Execution draft of deception attack inferred from network footprint (b) Deception attack executed*

and gateway through the host computer. Now the host computer fires unauthorized change of frequency command to VFD.

Step 3: Next action depicts response that will reach SCADA server due to the unauthorized change in VFD frequency. Response packets from gateway are crafted to standard frequency. The challenge is, to keep a check on the integrity and validity of the received data. The server assures validity response by comparing the transaction identifier of the reply and query. This check was breached by forming a counterfeit response that crafts in real time with transaction identifier reproduced from the corresponding query.

The process works within the constraints of Time-To-Live(TTL) of a TCP packet making it difficult to detect. Fig. 4.2 b illustrates the final snapshot execution of deception attack. The SCADA HMI developed in iVisionmax depicting operators panel that controls a speed of induction motor VFD. After execution of the deception attack, as expected operator's SCADA HMI view has the frequency of '0 Hz'(false/deceptive value), while induction motor was rotating with speed rated for the frequency of '25 Hz'. Right corner Fig. 4.2 b shows the reading on display panel mounted on VFD.

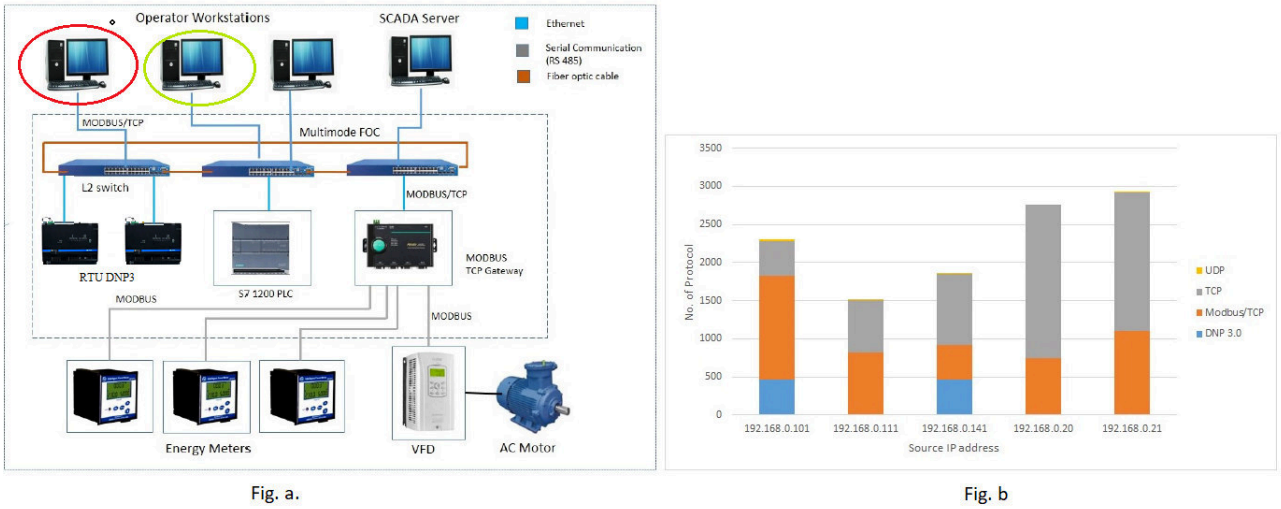


FIG. 4.3. *(a) Modified lab set up; (b) Packet analysis for modified lab set up*

4.2. Mitigation of Modbus Attack. The major vulnerabilities of Modbus/TCP protocol identified are as follows :

- If Modbus/TCP data is modified it does not affect the TCP checksum of the packet, hence making it easy to compromise the integrity of the data.
- Modbus/TCP protocol has no encryption, due to which packets can be easily modified by analyzing previous packets captured.
- A Modbus/TCP slave interprets the Modbus/TCP data and does authenticate master, making it vulnerable.

The above vulnerabilities are related to Modbus/TCP protocol implementation. However, this work does not try to mitigate protocol vulnerabilities.

We propose a systemic view, where in detection of concentrated Modbus/TCP protocol can be deceived. Fig. 4.1(b) shows packet analysis for network help to identify Modbus/TCP communication and devices involved. Fig. 4.3(a) shows modified set up, where in 'Green' circle marked to operator machine indicate it has been added with Modbus/TCP simulator with slave functionality and 'Red' indicates Modbus master.

Now, if we gather footprints of network as shown in Fig. 4.3(b), identifying SCADA server and gateway becomes difficult, since we have neutralised Modbus/TCP traffic over the network. Further if discovery phase is not conclusive it becomes difficult to carry out targeted attack.

In future, the machines simulating modbus/TCP can monitored for ARP spoofing that may lead to detection of attacker.

5. Conclusion and future work. Work presented provides an end to end methodology for security analysis which involves formal analysis using state space exploration of Coloured Petri net and Formal Concept Analysis along with the cyber-physical platform for testing. Deception attack for Modbus/TCP was modeled formally using CPN tool, further this representation of attack model was tested and verified on the in-house developed cyber-physical test bed. We have presented a unique approach of formalizing security concept of normal behavior and behavior under deception attack using formal concept analysis. In future work, we will explore the introduced novel framework of formal analysis and testing of security for wide-area monitoring, protection and control (WAMPAC) solutions.

REFERENCES

[1] R.Rajkumar, I.Lee, L.Sha, and J.Stankovic, *Cyber-physical systems:the next computing revolution*,Proceedings of the 47th Design Automation Conference. ACM, 2010, pp. 731736.

[2] K. Kim and P.R. Kumar, *Cyber-Physical Systems: A Perspective at the Centennial*,Proceedings of IEEE, 2012, pp. 1287-1308.

[3] I. Lee and O. Sokolsky,*Medical cyber physical systems*,Proceedings of 47thACM/IEEE Des. Autom. Conf., Jul. 2010, pp. 743748.

[4] Sridhar, S., Hahn, A., Govindarasu, M., *CyberPhysical System Security for the Electric Power Grid*,Proceedings of the IEEE , vol. 100, no. 1, Jan.2012, pp. 210-224.

[5] Genge, B., Siaterlis, C.,*Developing cyber-physical experimental capabilities for the security analysis of the future Smart Grid*,Innovative Smart Grid Technologies (ISGT Europe), 2011 2nd IEEE PES International Conference and Exhibition on , pp. 1-7, 5-7 Dec. 2011.

[6] S. Zonouz, K. M. Rogers, R. Berthier, R. B. Bobba, W. H. Sanders and T. J. Overbye, *SCPSE: Security-Oriented Cyber-Physical State Estimation for Power Grid Critical Infrastructures*, IEEE Transactions on Smart Grid, vol. 3, no. 4, pp. 1790-1799, Dec. 2012

[7] Y. Mo et al., *CyberPhysical Security of a Smart Grid Infrastructure*, Proceedings of the IEEE, vol. 100, no. 1, pp. 195-209, Jan. 2012.

[8] Gawanmeh, Amjad, Adel Bouhoula, Sofiene Tahar, *Rank functions based inference system for group key management protocols verification*, International Journal of Network Security 8, no. 2 (2009): 187-198.

[9] Mehr, Nima Sharifi, Christopher Dunn, Alexis Floyd, David James Kane-Parry, Volker Helmut Mosthaf, and Christopher Gordon Williams, *Security verification by message interception and modification*, U.S. Patent Application 15/422,253, filed February 1, 2017.

[10] Hollick, Matthias, Cristina Nita-Rotaru, Panagiotis Papadimitratos, Adrian Perrig, and Stefan Schmid, *Toward a Taxonomy and Attacker Model for Secure Routing Protocols.*, ACM SIGCOMM Computer Communication Review 47, no. 1 (2017): 43-48.

[11] S. Liu, B. Chen, T. Zourntos, D. Kundur and K. Butler-Purry , *A Coordinated Multi-Switch Attack for Cascading Failures in Smart Grid*, IEEE Transactions on Smart Grid, vol. 5, no. 3, pp. 1183-1195, May 2014

[12] G. Ericsson , *Toward a framework for managing information security for an electric power utilityCIGR experiences*, IEEE Trans. Power Del., vol. 22, no. 3, pp. 14611469, Jul. 2007.

[13] Chee-Wooi Ten, Chen-Ching Liu, Manimaran, G., *Vulnerability Assessment of Cybersecurity for SCADA Systems*, IEEE Transactions on Power Systems, vol. 23, no. 4, pp. 1836,1846, Nov. 2008.

[14] Sridhar, S., Govindarasu, M., *Model-Based Attack Detection and Mitigation for Automatic Generation Control*, IEEE Transactions on Smart Grid, vol.5, no.2, pp.580-591, March 2014.

[15] Ashok, A.; Govindarasu, M., *Cyber attacks on power system state estimation through topology errors*, Power and Energy Society General Meeting, IEEE , pp.1-8, 22-26 July 2012

[16] J. Hong, C. C. Liu and M. Govindarasu, *Integrated Anomaly Detection for Cyber Security of the Substations*, IEEE Transactions on Smart Grid, vol.5, no.4, pp.1643-1653, July 2014

[17] Chee-Wooi Ten, Chen-Ching Liu, Manimaran, G., *Vulnerability Assessment of Cybersecurity for SCADA Systems*, IEEE Transactions on Power Systems, vol. 23, no. 4, pp. 1836,1846, Nov. 2008.

[18] Matthew E. Luallen, *SANS SCADA and Process Control Security Survey*, A SANS Whitepaper, February, 2013 Available at : https://www.sans.org/reading-room/analysts.../sans-survey-scada-2013.

[19] H. Luiijf and R. Lassche, *SCADA (on)veiligheid: Een rol voor de overhead*, TNO/KEMA report,[Unclassified](June 2006),

[20] M. Naedele and D. Dzung , *Industrial information system security IT security in industrial plants An introduction*, ABB Review, issue 2, pp.6670, 2005.

[21] T. Smith, *Hacker jailed for revenge sewage attacks*, The Register, October 31, 2001.

[22] J. Visser, M. Berkom, J. Spiekhout, Y. Suurenbroek, J. Wessels, B. Smolders and C. Pietersen, Storing Gasmengstation, *Faults in Gas Mixing Stations*, Technical Report CB-2-02.060, 2002.

[23] Christiansson, Henrik, and Eric Luiijf, *Creating a European SCADA Security Testbed*, International Conference on Critical Infrastructure Protection. Springer US, 2007.

[24] E-ISAC, *Analysis of the Cyber Attack on the Ukrainian Power Grid*, https://ics.sans.org/media/E-ISAC SANS Ukraine DUC 5.pdf, Website last accessed June 24, 2016.

[25] The Modbus Organization, *Modbus Messaging on TCP/IP Implementation Guide V1.0a*, Modbus Organization: Hopkinton, MA, USA, 2004. 24

[26] The Modbus Organization, *Modbus Messaging on TCP/IP Implementation Guide V1.1a*, Modbus Organization: Hopkinton, MA, USA, 2004. 24

[27] Derrick and John., *Concurrent and Realtime Systems: The CSP Approach. By Steve Schneider*, Published by John Wiley and Sons Ltd., Chichester, UK, 2000. ISBN: 0471623733, 510 pages.

[28] Ryan, Peter, and Steve A. Schneider , *The modelling and analysis of security protocols: the csp approach* , Addison-Wesley Professional, 2001

[29] K. Jensen, L. Kristensen, and L. Wells, *Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems*, International Journal on Software Tools for Technology Transfer (STTT), vol. 9, no.3, pp. 213?254, 2007.

[30] Kurt Jensen, *Coloured Petri nets: A high level language for system design and analysis*, Springer, 1991.

[31] M.R. Hansen and H. Rischel, *Functional Programming in Standard*ML. Lecture Notes, 1997.

[32] Robin Milner, Mads Tofte, Robert Harper, and David MacQueen, *The definition of standard ml* , revised edition. MIT

Press, 1(2):23,1997.
[33] Wille, Rudolf, *Formal concept analysis as mathematical theory of concepts and concept hierarchies. ml*, Formal concept analysis 3626 (2005): 1-33.
[34] Ganter, Bernhard, and Rudolf Wille, *Formal concept analysis: mathematical foundations. ml*, Springer Science & Business Media, 2012.
[35] Priss, Uta, *Formal concept analysis in information science. ml*, Arist 40.1 (2006): 521-543.
[36] *Serhiy A. Yevtushenko*, *System of data analysis Concept Explorer*,(In Russian). Proceedings of the 7th national conference on Artificial Intelligence KII-2000, p. 127-134, Russia, 2000.
[37] C. B. Vellaithurai; S. S. Biswas; A. K. Srivastava ,*Development and Application of a Real-Time Test Bed for Cyber-Physical System*, IEEE Systems Journal ,no.99, pp.1-12 36
[38] B. Chen, N. Pattanaik, A. Goulart, K. L. Butler-purry and D. Kundur , *Implementing attacks for modbus/TCP protocol in a real-time cyber physical system test bed*, IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR), Charleston, SC, 2015, pp. 1-6. 36
[39] Y. Yang et al, *Cybersecurity test-bed for IEC 61850 based smart substations*, IEEE Power & Energy Society General Meeting, Denver, CO, 2015, pp. 1-5.
[40] V. Venkataramanan, A. Srivastava and A. Hahn , *Real-time co-simulation testbed for microgrid cyber-physical analysis*, Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES), Vienna, Austria, 2016, pp. 1-6.
[41] U. Adhikari; T. MORRIS; S. Pan , *WAMS Cyber-Physical Test Bed for Power System, Cybersecurity Study, and Data Mining*, IEEE Transactions on Smart Grid
[42] Stouler, Keith, Joe Falco, and Karen Scarfone , *Guide to industrial control systems (ICS) security*, NIST special publication 800.82 (2011): 16-16.

Edited by: Amjad Gawanmeh
Received: May 5, 2017
Accepted: Oct 30, 2017

DOI 10.12694/scpe.v18i4.1332
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 18, Number 4, pp. 331–346. http://www.scpe.org

MIDDLEWARE CHALLENGES FOR CYBER-PHYSICAL SYSTEMS

NADER MOHAMED* JAMEELA AL-JAROODI† SANJA LAZAROVA-MOLNAR‡ AND IMAD JAWHAR§

Abstract. Cyber-Physical Systems (CPS) are being developed to provide useful interactions between physical systems and environments and cyber world for a variety of applications. CPS are designed with a set of software and interconnected distributed hardware components that are linked with physical elements to provide advanced monitoring and control mechanisms geared towards enhancing the targeted physical system or environment. These components function seamlessly to offer specific functionalities that help enhance human lives, physical system operations and environments. While CPS can offer many smart enhancements for improving physical processes, the development of such complex systems composed of many distributed and heterogeneous components is extremely difficult. This is due to the many communication, computing, and networking challenges. Using an appropriate middleware that provides a framework to support developing and operating diverse CPS applications is a novel method to address these challenges. The availability of advanced middleware services and platforms can provide effective approaches for enhancing CPS application development processes as well as provide more robust environments for operating CPS applications. Such middleware can significantly reduce the time needed to design, build, test, and operate robust CPS applications. However, designing a common middleware platform for diverse types of CPS applications is not trivial. This paper investigates the middleware challenges for CPS, based on the different types of CPS applications being developed and their specific challenges. In addition, the paper discusses the current efforts of developing middleware platforms for CPS and the open research issues in the field.

Key words: Cyber-physical systems, Middleware, Software engineering, Application development.

AMS subject classifications. 68U99

1. Introduction.

There are many CPS applications that add enhancements and smart features to several types of physical systems and environments. CPS can add smart mechanisms to fully automate manufacturing processes, manage and enhance the operations and safety of environments and infrastructures, and enable Unmanned Autonomous Vehicle (UAV) operations and applications. They can also enhance the safety of transportation systems, enhance energy consumption in smart buildings, and improve healthcare for patients. CPS combine various concepts and technologies from embedded systems, networks, distributed systems, software, and hardware; as well as other engineering disciplines such as systems, mechanical, control, civil, and electrical engineering to provide added features to the physical world [67].

While CPS can offer many smart enhancements for improving physical systems and processes, the development of such complex systems composed of many components interacting in various ways and capabilities is extremely difficult. CPS attach different hardware components like sensors, actuators, microcontrollers, and other devices to physical systems or environments and use distributed software that implements smart algorithms to control the corresponding physical system [48, 35]. The distribution and heterogeneity of these devices and their links with the physical components make the design, development, and operations of CPS very challenging.

Due to the importance of CPS applications and the complexity of their development process, huge research efforts started investigating the different issues associated with CPS and their applications. These include security, reliability, performance, quality, validation, and development methodologies and tools [48]. In our previous work, we discussed the importance of software components in any CPS and highlighted the main software engineering issues that include the complexity of analysis, design, development, and testing for CPS software [20]. We also briefly highlighted different research directions to tackle these issues. One of these directions is to develop and use advanced middleware platforms to support CPS applications. In this paper, we investigate in detail the middleware challenges for CPS. This investigation is based on the different types of CPS applications and their specific challenges. This paper can help CPS applications developers to recognize the middleware challenges and requirements of different CPS applications. As a result, suitable approaches can be selected and adapted to fit the specific needs of the applications being considered. Current and new approaches

*Middleware Technologies Lab., Pittsburgh, Pennsylvania, USA (nader@middleware-tech.net).
†Department of Engineering, Robert Morris University, Moon Township, Pennsylvania, USA (aljaroodi@rmu.edu).
‡University of Southern Denmark, Odense, Denmark (slmo@mmmi.sdu.dk).
§Midcomp Research Center, Saida, Lebanon (imad@midcomp.net).

for middleware platforms need to be considered, yet most of the current approaches may not fit well with the nature of CPS applications. For example, a middleware platform such as real-time CORBA has shown promise for general distributed systems with some constraints; however, it lacks the flexibility in dealing with the CPS challenges.

Middleware challenges for related systems such as Wireless Sensor Networks (WSNs) and the Internet of Things (IoT) were thoroughly investigated [70, 60, 68, 25]. Furthermore, several papers investigated and discussed general challenges in CPS [48, 35, 74]. However, none investigated in detail the middleware challenges for CPS. Identifying and studying middleware challenges is important to be able to design and develop the most suitable middleware platform to support different CPS applications. In our previous work [58], we briefly highlighted how middleware can provide different types of support for CPS. In this paper, we extend our previous work including detailed middleware challenges, current research efforts in this regard, and discuss several open research issues that need to be investigated and resolved to offer the best possible middleware platforms for CPS.

In the rest of the paper, we offer some background about CPS and middleware in Section 2. In Section 3, we discuss different CPS applications to understand the role middleware can play to support them. We then discuss and identify the general and common challenges of CPS applications in Section 4. Section 5 discusses the role of middleware to support CPS applications while middleware challenges for CPS are identified and discussed in Section 6. The current research efforts for CPS middleware are discussed in Section 7 and Section 8 lists some open research issues in the field. Finally, we offer some concluding remarks and future directions in Section 9.

2. Background. Here we provide preliminary information about CPS and middleware. This establishes a base line for the upcoming discussion of middleware platforms for CPS.

2.1. Cyber-Physical Systems. CPS are distributed embedded systems developed to support smart and context-aware mission-critical applications in different domains such as energy, manufacturing, healthcare, civil infrastructures, automotive, transportation, aerospace, entertainment, and consumer appliances. CPS provide monitoring and control functions to help achieve specified goals to benefit the application domain. Unlike traditional embedded systems, CPS are complex embedded systems with distributed components and processing capabilities. Embedded computing devices of CPS can be in sensors, actuators, microcontrollers, and other devices usually connected with a wired or wireless network and tightly coupled with their physical environment.

As CPS are embedded in physical environments, they provide useful interactions between the computational and physical elements through intelligent mechanisms. These intelligent mechanisms can be organized in four main steps:

- Observing the status of the physical system or environment using different type of sensors that are attached to the elements of the physical system or environment.
- Building a knowledgebase about the physical system environment from the collected sensed information using software functions and storage systems.
- Making decisions to enhance or control the physical system or environment to meet specific objectives defined for the application. This is done using the knowledgebase and some status information of the physical system or environment with the help of smart algorithms that run on some integrated computing components.
- Applying the enhancement or control actions using actuators that are attached to the elements of the physical system or environment.

These four steps are linked in a closed loop as shown in Figure 2.1 to allow the CPS to provide full monitoring and control functions to achieve the needed objectives. These objectives can be to provide the adaptability, autonomy, efficiency, functionality, reliability, safety, and usability of the system or environment.

2.2. Middleware. Middleware for a distributed system is a logical software layer that abstracts the details of the underlying distributed components and provides a set of services to develop and operate distributed software applications beyond those available from the individual components. The provided services can be used to support integration, system management, runtime for distributed code, fault tolerance, security, or

FIG. 2.1. *Closed-loop control steps of CPS*

load-balancing. In addition, it can provide more advanced services to support issues related to power consumption, limited resources devices, etc.

Middleware has become a necessary part of distributed computing. It is practically impossible to develop large-scale distributed systems or applications without involving middleware [18]. By comparison, trying to build a distributed systems or application without middleware is like trying to write a simple application on a personal computer without the operating system. Generally, the main functions of different middleware platforms are [18]:

- Providing tools to simplify the development of complex distributed applications.
- Providing high-level abstractions and interfaces to facilitate distributed application integration and reuse.
- Hiding the heterogeneity of the underlying environments.
- Hiding the distribution and communication details in the underlying environment.
- Enabling communications among the different distributed components of the infrastructure.
- Offering services for common functions needed by different applications to reduce development and duplication efforts.
- Providing a common architecture to enable adding new services and features without having to change the distributed applications.
- Providing added-value features and nonfunctional properties such as security, reliability, scalability, and Quality of Services (QoS).

Due to the advantages of using middleware, several middleware platforms were developed for different distributed applications and environments. Examples include WSNs [66], cloud computing [78], collaborative UAVs [57], and mobile social networks [38]. The developed middleware solved many issues in these environments and similarly can solve many issues in CPS.

3. CPS Applications. As CPS can provide useful interactions between physical and cyber worlds, a number of domains such as health, energy, transportation, and security can greatly benefit from CPS applications [35]. However, developing and operating such applications can face with many challenges. To identify and understand these challenges, we discuss some important CPS applications used or proposed for different application domains. We highlight their benefits as well as their development and operations challenges. This will help us identify the type of support needed by the middleware platforms designed for CPS applications.

In the energy domain, CPS are used to add values such as efficiency, reliability, and sustainability of the production and distribution of electric power in smart grids [43]. A smart grid is a renovated electrical grid system that uses information and communication technology (ICT) to collect and act on available information on the behaviors of suppliers and consumers in an automated fashion. A smart grid is a CPS that provides self-monitoring and advanced control mechanisms for power production and distribution, as well as addressing consumer needs towards increased grid efficiency and reliability. This involves placing smart sensors and meters on production, transmission, and distribution systems in addition to consumers locations to get granular near real-time data about the current power production, consumption, and faults. Although the smart grid has many potential benefits, it requires the collection and analysis of huge amounts of data continuously. This collected data is processed in real-time to send back control information to adjust the operational conditions and improve efficiency, reliability, economics, and sustainability of the system. In addition, the processes

used to generate renewable energy from hydropower plants [56] and wind power plants [75] are controlled by CPS. Furthermore, the energy consumption in smart buildings is controlled and monitored by CPS [36]. The buildings equipment such as HVAC (Heating, Ventilating, and Air-Conditioning) systems, appliances, and lighting systems are controlled with CPS. Smart buildings CPS are usually equipped with different types of sensor nodes that monitor the current energy usage and environmental conditions. These sensors report their observations and measurements to a centralized CPS monitoring and control system. The control system implements intelligent algorithms to control the systems used in the buildings to optimize energy use based on the sensed observations and current operational and environmental conditions. CPS also provide control mechanisms for energy efficiency in data centers [79, 62].

In the health domain, many medical systems are controlled by CPS. Medical CPS should provide safe and intelligent continuous care for patients [49]. Medical CPS are networks of medical sensor devices that provide clinical monitoring functions such as heart-rate and oxygen levels; medical delivery devices such as infusion pumps and ventilators; and control devices that provide the main controls for medical CPS and are responsible for efficient and safe operations of the whole system. As medical CPS are life-critical systems, they must be context-aware, reliable and resilient to faults. Some medical CPS can be wearable systems and continuously used by patients. These wearable systems usually operate on battery power. In this case the medical CPS must be energy-efficient and designed to minimize energy consumption to extend the life of the devices.

In the transportation domain, an important CPS application area that recently received high attention is the vehicular safety applications. There are many safety applications for vehicles including lane change warning messages, emergency breaking, collision avoidance mechanisms, and blind spot monitoring. These applications provide fully automatic or semi-automatic actions to enhance driving safety. Some of these applications are based on individual vehicle observations, decisions, and actions while others are based on collaborative observations, decisions, and actions where neighboring vehicles exchange messages for that purpose [53]. The various devices and components needed to achieve these functionalities form the vehicular CPS [30]. Embedded software is used to implement different safety applications. The most important features of vehicular safety applications are the real-time and reliability support in detection and response. All aspects of vehicular safety applications including threat observations, decision making, communication, and actions must be in real-time and reliable. This imposes a serious restriction on how the software is designed and how well it supports high levels of integration across all the devices involved to ensure real-time and reliable responses. In addition, self-driving cars are considered as CPS [24]. Since they practically integrate all the mentioned features in addition to vision and monitoring components to allow the vehicle to navigate the roads based on sensed data and intelligent software that interprets and responds to this data in real-time. Other transport CPS include intelligent traffic light controls which include monitoring devices across multiple locations to accurately predict traffic patterns and adjust traffic lights to optimize flow. One example of such domain is discussed in [71].

In the security domain, CPS can be used to monitor and protect important large-scale infrastructures such as long oil and gas pipelines that extend for hundreds to thousands of miles [22]. These usually extend across unattended and sometimes difficult areas such as underwater, deserts and forests. Pipelines are considered important infrastructures that need to be monitored and protected as they provide the main supplies of energy and water for many countries and areas. Sensors, actuators, and other devices are usually deployed and connected using wired or wireless networks to monitor, control, and protect such infrastructures. In addition, CPS can be used to protect water networks and to make them smarter, more efficient, more reliable, and more sustainable. CPS can be embedded within water networks to involve some monitoring and control mechanisms and to add smart features to the operations of water distribution [44]. One of these functions is to provide early warning mechanisms to identify problems in water networks. For examples leaks and pipe bursts can be easily detected while fast and temporary solutions can be applied to reduce water waste and to minimize further risks or damages to the network.

Other CPS applications include greenhouses efficient control that aims to provide efficient control for suitable climate, soil, lighting, and water level in greenhouses [32]. Similarly, they can be used to enable smart homes [41]. In addition, CPS are used to autonomously operate unmanned vehicles CPS provide networks that connect the payloads on these vehicles like sensors, actuators, cameras, storage, communication devices, and the microcontrollers of the vehicle [57]. CPS are also used to automate, control, monitor, and enhance manufacturing

TABLE 3.1
CPS Applications and their Cyber and Physical Worlds

CPS Applications	Physical Parts	Cyber Parts	Benefits
Medical CPS	Patients, illnesses, and drugs	Monitoring and controlling patient health status and data	Timely and accurate patient monitoring and treatment
Smart Buildings	Buildings, temperature, lighting, air quality and building residents	Monitoring environmental conditions and energy usage and implementing algorithms to control equipment	Reducing energy consumption and maintaining required quality of living
Smart Grid	Electricity, fuel, power generators, power distribution networks, and consumer devices	Real-time monitoring and controlling energy productions and consumptions	Optimizing energy utilization, reducing overload risks and energy waste
Gas & Oil Pipelines Monitoring & Control	Oil, gas, pipeline networks, pipeline physical status including pressure, and temperature	Monitoring the pipeline status and controlling the pipeline operations	Maintaining health and operations of the pipeline and reducing the impact of failures and accidents
Smart Water Networks	Water, water distribution networks and their status, water storage, water pumps, and water generators	Monitoring and controlling the process of transferring and storing water and its quality and its usage	Reducing water loss, optimizing water production and utilization, and enhancing water quality
Vehicular Safety Applications	Vehicles, vehicles mechanical and electronic devices, roads, drivers and passengers	Real-time algorithms to monitor vehicles status and control vehicles to avoid accidents and optimize flow	Reducing the possibility of accidents, congestion and traffic violations
Intelligent Traffic Light Controls	Traffic lights, vehicles and their positions, roads, pedestrians, and intersections	Real-time algorithms to monitor traffic status and to control traffic lights	Reducing traffic delays, minimizing vehicles travel times, increasing vehicles average velocity, and enhancing the prioritization for emergency vehicles movements
Self-Driving Vehicles	Vehicles equipment, Vehicles energy, passengers, resources, position, roads, roads traffic lights and signs, and neighboring vehicles	Smart algorithms to automate vehicle driving and to maintain driving safety	Reduce transportation costs, optimize traffic flow and enhance safety
Manufacturing Control and Monitoring	Manufacturing requirement, raw materials, products, workers, and warehouses	Real-time algorithms to monitor and control production processes, equipment and material utilization, and product quality	Optimize production and maintenance and enhance product quality
UAV	UAV equipment and energy, position, and operational space	Algorithms to control the UAV, optimize movement and collaboration	Enhanced operations and safely and achieving operational goals
Energy Efficiency in Data Centers	Data center equipment, energy supplies including renewable energy, equipment and buildings temperatures, ventilation and air conditioning	Algorithms to monitor the status of the data center and control servers operations and temperatures in the center	Reducing energy consumption and maintaining the good health of the equipment
Wind Farms	Wind, wind turbines, control equipment, and energy distribution and storage equipment	Algorithms to monitor and control wind turbines and the produced power and to optimize energy production and storage	Maximizing power generation and enabling integration with other systems such as smart grids
Hydropower Plants	Hydraulic engines, transducers, power meters, electric power generators, water level, water flow, energy storage and distribution equipment	Algorithms to monitor and control the power generation, distribution and storage, and the water flow	Maximizing power generation and enabling integration with other systems such as smart grids
Greenhouse Efficient Control	plants, climatic conditions, soil, ventilation, carbon dioxide, water, and heating, cooling, and ventilation equipment	Algorithms to regulate greenhouse climate, optimize resources utilization and maximize production	Enhancing plants growths and produce production and quality, and optimizing resources (e.g. water and energy) utilization

processes [50]. Table 3.1 summarizes the CPS applications in terms of their physical and cyber parts and their benefits.

4. CPS Applications Challenges. We realize based on the discussion of the various applications of CPS in Section 3 that there are some common challenges facing developing these applications. The main challenges include:

1. Real-Time operations: Most CPS applications need to function in real-time to deliver usable information. This includes real-time sensing, communication, processing, decision making, and actions. In most applications, the earlier we receive status information and generate the required controls, the better

results we can achieve. For example, a self-driving vehicle, requires immediate knowledge of the route changes, traffic conditions and traffic light status. Any delays may lead to catastrophic results such as not being able to respond correctly to a changing traffic light or another vehicle not behaving correctly.

2. Heterogeneous Devices: CPS applications are built with multiple heterogeneous devices like sensors, actuators, microcontrollers, and communication and storage devices. In addition, they operate in heterogeneous physical systems and environments. This leads to complications in implementing the various controls and integrating the different components as each will need its own models and software components. As a result, introducing new equipment or changing current ones, which occur frequently, will have to involve changes in the software being used.
3. Limited Capability Devices: Some CPS need to use devices with limited capabilities and remote functions. This is mainly due to the current limitations on available devices or to reduce the cost of the CPS. These devices usually have limited wireless communication, processing, and storage capabilities and many also may have limited power sources. Careful design is needed to include these devices as components CPS. This will require complex algorithms to manage, operate and control these devices within their limitations.
4. Distributed Processing: CPS applications require distributed processing and decision making to enhance their operations. In addition, some applications need to use parallel processing for faster operations. This introduces three major challenges: one is the different types and models of communication to be used given the heterogeneity of the devices and connections being used; another is the delays and reliability of the communication, which must be addressed efficiently to enable stable and reliable operations; in addition to the security and privacy of the system and information being used as they travel over various communication channels in the system.
5. Security and Privacy: As most CPS support distributed critical applications, there are high security and privacy concerns. These are introduced mainly by the distributed nature of the system and its components. In addition, many CPS are used within the context of critical and private domains, where data must be protected for various reasons. Therefore, the security and privacy of the information and software must be protected.
6. Reliability and Fault Tolerance: Many CPS applications are critical applications; therefore, they need to be reliable and highly available. It is important to implement mechanisms to increase reliability, which vary depending on the types of devices and software being used and the operational parameters of these devices. In addition, data integrity and correctness must be preserved to achieve high reliability of the system. These systems should be able to operate effectively even when faults occur and should be able to detect and resolve different types of faults without negatively impacting the physical systems.
7. Communication: Special communication requirements and capabilities are needed by some CPS applications among their devices and subsystems. These requirements may include real-time support, highly available and reliable communication, high-bandwidth and efficient information exchange. Some large-scale and highly distributed CPS applications may also need optimized communication mechanisms for collecting and distributing data among the systems components. In addition, different components of a CPS may require different modes of communication and others could be designed to adapt the communication modes to available resources.
8. Mobility: Some CPS applications involve mobile devices that need to be efficiently and securely connected with the rest of the system. This may require actively managing the mobile devices as they change location. Thus, the software used should be able to discover, monitor and control mobile devices are they travel within, enter or exit the geographic boundaries of the system.
9. CPS Devices Locations: CPS applications that involve mobile devices, need to be aware of the mobile devices locations to optimize their operations and correctly achieve the applications objectives. For many CPS applications, knowledge of the instantaneous location of the mobile devices help optimize operations using these devices. For example, when a UAV is available in a specific area, it could be given a specific task relevant to that area to accomplish before moving out of range.
10. Power Limitation: Some CPS components are used in remote locations or in areas where no constant power sources are available. These devices can operate for a limited time and may not be easily

replaceable. Thus, the CPS design should focus on extending the life of these components using efficient and power-aware software, hardware, and communication protocols. The software needs to optimize their operations and limit access to their resources using other alternatives when possible.

11. Integration with Other Systems: Some CPS applications require integration with other computing systems or other CPS. For example, integrating CPS applications in neighboring vehicles to enable collaborative safety applications across multiple vehicles or integrating some components with powerful resources and platforms such as cloud-based services offering processing power, storage, and data services. This extends some of the challenges discussed such as security and privacy, reliability and real-time support in addition to the introduced complexity of ensuring seamless interoperations between the systems.
12. Intelligent Decisions: Most CPS applications need to make intelligent decision to optimize their operations. This may involve including intelligent algorithms such as data mining algorithms in the CPS design. Many of these require large amount of data and intensive computations, as a result, efficient mechanisms to enable these operations need to be included in the application and could also lead to the need for integration with more powerful systems such as the cloud to support these operations.
13. Context Awareness: Some CPS cannot properly or efficiently function without knowing the context of their systems resources, physical environment and general domain. This will require the knowledge of specific information such as power levels, communication and processing status, and other external physical contexts utilized for the operations of CPS. Therefore, the software used must be able to collect context data, organize it and make it available for the CPS application to be used effectively.
14. Big Data: Some CPS have a large number of sensors that continuously generate data resulting in huge data sets [80]. At the same time, these systems cannot be optimized well to achieve the objectives of the applications without analyzing this collected data. This may require having a good system infrastructure capable of dealing with the challenges of storing and processing big data as well as algorithms capable of the required analysis.

Table 4.1 summarizes the challenges for different CPS applications. These challenges make developing CPS applications without using proper tools and middleware very difficult. Developers need to be aware of the specific challenges for the application being developed. They need to find effective and efficient methods and algorithms to address these challenges and implement them within the software. As a result, the application must include complex components addressing the actual domain of the application in addition to the challenges imposed by the CPS environment, architecture and components.

As we can see not all challenges apply to all CPS, and for this reason, we identify the challenges that apply to CPS with respect to their application domains. Table 4.1 shows the different application domains of CPS and what challenges are of importance to that domain. As an example, we use smart buildings as an emerging type of CPS. Smart buildings feature Building Management Systems (BMS) that provide monitoring and control to the various elements and subsystems, in fulfillment of preset objectives. Typical objectives of smart buildings are minimizing energy consumption and maximizing occupants comfort. The challenges relevant for smart buildings are the following: Real-time operations, as buildings host occupants, whose well-being depends on the non-stop correct operations of the system; Heterogeneous Devices, as smart buildings incorporate a variety of devices to support their daily functions [64] ; Security and Privacy Support, as the security and privacy of occupants data regarding their interactions with the buildings systems need to be protected against any threats or attacks and must be kept private [23] ; Reliability and Fault Tolerance, as some buildings are safety-critical (e.g. hospitals or power stations) where the probabilities of failures should be minimized [47] ; Integration with Other Systems, which is especially important when buildings are considered elements of smart cities and smart grids, where they will need to function flawlessly in connection to other systems (e.g. traffic or smart grid, but also other smart buildings BMS); Intelligent Decisions, as BMS goals are to optimize buildings performance; and Context Awareness, as smart buildings are not isolated and their environment plays a vital role in their performance [33].

5. Middleware for CPS. Middleware platforms can provide important support for developing and operating CPS applications. The availability of a suitable middleware platform that provides solutions for the challenges discussed earlier will not only enhance the CPS development process, but also enhance reusability

TABLE 4.1
Different Challenges of Different CPS Applications in the General Cases

-	Real-Time Operations	Heterogeneous Devices	Limited Capabilities	Distributed Processing Need	Security and Privacy Requirements	Reliability & Fault Tolerance	Special Communication	Mobility Challenge	Location-Based Requirement	Power Limitation Challenge	Integration with Other Systems	Intelligent Decision Requirement	Context Awareness Challenge	Big Data
Medical CPS	x	x	x		x	x	x	x		x	x	x	x	x
Smart Buildings	x	x			x	x					x	x	x	x
Smart Grid	x	x		x	x	x	x				x	x	x	x
Gas & Oil Pipelines Monitoring & Control	x	x	x	x	x	x	x			x	x		x	
Smart Water Networks	x	x	x	x	x	x	x			x	x	x	x	x
Vehicular Safety Applications	x	x			x	x		x	x			x	x	
Collaborative Vehicular Safety App.	x	x		x	x	x	x	x	x		x	x	x	
Self-Driving Car	x	x		x	x	x		x	x		x	x	x	
Manufacturing	x	x			x	x					x		x	x
UAV	x	x	x		x	x		x	x	x		x	x	
Collaborative UAVs	x	x	x	x	x	x	x	x	x	x	x	x	x	
Energy Efficiency in Data Centers	x	x		x	x	x			x	x	x	x	x	
Wind and Hydro Power Plants	x	x		x	x	x					x		x	
Greenhouse Efficient Control		x	x		x	x				x		x	x	

and maintainability, reduce risks, and reduce the overall cost of developing and maintaining CPS applications. With a suitable middleware platform designed to support CPS applications, the developer can focus more on developing the main functions of the CPS applications rather than expending huge time and effort in implementing solutions and codes to solve the general and common issues such as reliability and security. The specialized middleware for CPS can be designed to include a set of services that provide solutions for various common CPS challenges that exist for any type of CPS applications. These services can be used by developers to implement different CPS applications. The developers can use middleware APIs for these services to utilize their functions and features and integrate them with the required specific functions of the CPS. In addition, the middleware can enable reuse of any previously implemented module in new CPS applications. This will also reduce the time and efforts needed to test new modules as the reusable modules have already been tested and approved for use. The middleware will also enable maintainability by allowing easy changes for any modules in the system. Any software module or service can be changed or replaced by better implemented one if it uses the same interface. This will also enable easy change to some hardware components in CPS applications to better hardware components.

The architecture of CPS implemented with middleware has five layers as shown in Figure 5.1. These layers are physical environment layer, observation and action layer, network layer, middleware layer, and application layer. The physical layer is the lowest layer in the CPS architecture and it consists of all the physical environment or parts of the CPS that will be monitored or controlled by CPS applications. This layer can include parts like vehicles, streets, buildings, the human body, machines, energy, etc. The second layer from the bottom is the observation and action layer. This layer is responsible for observing the current status of the physical environment and acting on the environment by changing its current state. This layer will have different types of sensors such as temperature sensors, movement sensors, cameras, sound sensors, pressure sensors, GPS, RFID, bar code readers as well other types of sensors and monitors to observe the current status of the monitored environments. In addition, it also has a number of actuators such as hydraulic, pneumatic, electric, thermal,

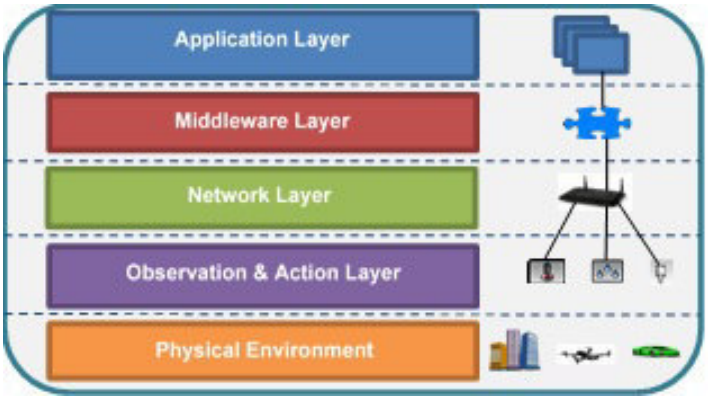


FIG. 5.1. *CPS Layers*

and mechanical actuators. These actuators are responsible for moving or controlling mechanisms, systems, or environments and changing their current physical states. The third layer is the network layer which is responsible for collecting the data observed by the observation and action layer and transporting it to the upper layer, the middleware layer. It is also responsible for transporting information from the middleware layer to the observation and action layer. The network layer can deal with simple peer-to-peer networks or with multi-hop networks of any scale. The network can be wired, wireless, or both. Generally, this layer is responsible for enabling information exchange between the middleware layer and the observation and action layer.

The Middleware layer, which is the focus of this paper, is responsible for providing a set of services to support implementing and operating CPS applications. These services can be basic communication services that enable the exchange of messages and information among CPS components or value-added services such as real-time support, action validation support, reliability and fault tolerance support, mobility support, location-based support, and security attacks detection services. In addition, these services can be very advanced smart services that are controlled by specifying high-level policies and global objectives for the whole systems operations. Examples of these services can be context awareness services, multiple CPS collaboration services, intelligent decision services, self-adaptive services, self-resilient services, and self-protected services. These middleware services can be selectively used by the top layer, the application layer to develop and operate CPS applications specific to the needs of the CPS in use. The Middleware layer can abstract the heterogeneity and distribution of the lower layers including the physical environments and hide their technical details. This feature can reduce the complexities of CPS applications. The CPS applications developers can have a set of advanced application interfaces (APIs) provided by the middleware to use the provided services. These services provide ready-made solutions for common challenges facing different CPS applications. The developer will not need to spend a significant time developing, implementing, and testing new code for these common challenges. As a result, using the right middleware for CPS can reduce the risks of having bugs in the CPS applications as the available middleware services will be developed more carefully and used by multiple applications. The middleware approach will allow for pre-designed services modeled specifically for the devices and environments in use, thus allowing for better utilization of these components. In addition, adding, changing or removing components will not require major changes in the application software as the middleware makes it possible to perform these tasks while maintaining a consistent API for the applications. From the developer perspective, a device performing a specific type of activity is merely a service available with known APIs to access it. The actual details of its operations and technical specifications are hidden and of no concern to the application developer. At a higher level, since we view the features needed as services, it becomes possible to implement new services, update current services, and provide alternative implementation of some services allowing the application developer to pick and choose the suitable services for the application being developed. Furthermore, this allows for updates, increments and adjustments to currently used services without having to rebuild the applications using them.

We classify middleware systems that can support CPS applications into three types. These three types are based on their abstraction levels and their support functions. Each of these types can support different

programming models to develop and operate CPS applications. Here discuss the main characteristics and functions of the three types and the relationships among them.

5.1. Communication Middleware for CPS Applications. This type of middleware mainly enables and facilitates communication among heterogeneous and limited resources CPS components. This includes offering services for efficient unicast communication and efficient group communication for broadcasting, multicasting, and data collection. It can provide basic security mechanisms to be used by developers to protect the communications among CPS components. It also provides basic mechanisms to communicate with other systems through message passing and/or remote procedure calls. While this type of middleware enables communication among CPS components, the developers need to solve and write codes for many other challenges mainly due to its limited scope and functions. The developers also need to handle the details of distributed processing in CPS. One possible approach for a communication middleware is to use a customized massing passing model which can meet the CPS communications challenges.

5.2. Value-Added Services Middleware for CPS. In this type, a number of services can be provided and used part of the CPS applications adding features and value to these applications. These services can be real-time support, monitoring, validation, reliability, fault tolerance, mobility support, location-based support, security attacks detection services, and service-level integration with other systems. A higher level of programming models for distributed and parallel processing can be used with this middleware to allow developers to implement CPS applications using the available services. This middleware type includes a resource manager and scheduler to enable implementing the services and also to allow developers to define available resources and detailed policies guiding the use of the available services. One of the most suitable middleware architectures to use here is the service-oriented middleware [19].

5.3. Advanced Services Middleware for CPS. Here we describe advanced services and smart support components, as an addition to the services described as Value-Added Services Middleware. These include autonomous resource discovery and management, context awareness support, multiple CPS collaboration support, and intelligent decision support. In addition, smart services such self-adaptive, self-resilient, and self-protected services can be offered. The developers in this case will resort to advanced high-level abstracted programming models to write the CPS applications instead of conventional programming languages. The developers can also specify high-level policies and global objectives for the whole system to operate on. Examples of these programming models in related systems such as WSNs are Kairos [34] and Cougar [81].

These three middleware types differ in the level of abstraction and how much time and effort needed by the developers to implement new CPS applications. The developers need more time and effort to develop new applications with the communication middleware alone as they need to deal with individual components in the CPS, while they need less time and effort with the value-added services middleware as they will use the available services to implement their applications. Furthermore, they need much less time with the advanced services middleware as they will deal with high-level policies and global objectives and the middleware will map these into implemented services. However, this type of middleware is very difficult to implement as it needs to self-handle most of the CPS common challenges.

6. Middleware Challenges for CPS. Although middleware platforms provide many advantages for implementing and operating CPS applications, developing such middleware platforms is also challenging. In this section, we discuss some of the challenges of designing and implementing such middleware platforms:

1. Enabling Smooth and Efficient Integration: CPS middleware should enable smooth and efficient integration among all CPS heterogeneous components. Components in any CPS can be developed and implemented by several manufactures. While some CPS components are implemented such that they support standard interfaces for interaction with other components, others are implemented without supporting any interface standards. One of the roles of the middleware is to enable the integration among these heterogeneous components and ensuring proper integration with the various available interfaces. In addition, these components may have different communication capabilities and operational standards. They may use different communication protocols, different communicate rates, different synchronization capabilities, and different security capabilities. The role of the middleware here is to enable the efficient and smooth integration among all used communication models in use and enabling seamless communication between the devices and components used by the CPS applications.

Furthermore, the CPS middleware should enable the integration with other systems outside the boundaries of the CPS environment such as cloud and fog computing and other CPS systems.

2. Supporting Advanced Communication Schemes: Some CPS applications cannot be efficiently offered without using advanced communication schemes such as the Publish/Subscribe communication scheme [29] which is needed for reducing communication overhead in large scale systems with a large number of sources of information or events such as large numbers of sensors and control components or actuators using the produced information and events. Another example of these advanced communication schemes is the store and forward communication scheme [45] which is needed for large mobile applications with discontinuous communication links among their mobile CPS components. These applications can be collaborative UAVs or collaborative vehicular applications. These advanced communication schemes are usually not supported by the current traditional communication technologies while they can be effectively offered by the middleware layer and used by the CPS applications.

3. Resource Management: The need for providing real-time, reliable, fault tolerant, power efficient, and automatic management by the CPS middleware require having an efficient and smart resource manager that can provide essential features that provision such services. These features include efficient recourse discovery, monitoring, and control for both limited and unlimited capabilities components in the CPS. In addition, the resource manager should be supported by an efficient scheduler for utilizing these resources as well as QoS support for both processing and communication. The middleware should be able to map the performance requirements of different applications into system level parameters that can configure the underlying system to achieve these specified performance requirements. However, designing the scheduler and mapping processes for CPS middleware is challenging given the large number of resources and the variety of specifications, functions and requirements of each of these resources. Moreover, the design of fast and optimal or near-optimal algorithms for resource allocation and adaptation can also be very challenging. This is due to heterogeneity of the resources, the limited capabilities of some of the CPS components, and the high CPS application requirements such as safety, security, scalability, and sustainability. Furthermore, the scheduler needs to make tradeoff decisions among communication, computations, monitoring, and control to achieve the CPS applications objectives. These tradeoffs can be challenging and requires smart decisions for optimal or semi-optimal resource scheduling.

4. Secure Middleware Services: As most CPS are considered critical applications and the middleware is the backbone for integrating the CPS components and enabling the CPS operations, then all middleware services should be secure. Any security leakage in the offered services can be utilized to gain unauthorized access to the CPS applications, which imposes many risks such as suspension the operations of the CPS applications or altering the operations of the CPS applications to unsafe operations thus resulting in damages in the corresponding physical environment or systems or interfering with normal operations leading to serious problems.

5. Global Reference Time Support: Operations in many CPS applications cannot be correctly done without having a global reference time to be used by all components of CPS to order and synchronize events and actions in the CPS [77]. This is one of the requirements in some CPS applications for ensuring safe and accrued operations. While a physical reference time support is offered in some new hardware and networks, it is not provided and supported by many CPS components. In this case, the CPS middleware can provide a logical global reference time support to be used for CPS applications. This requires designing accurate middleware-based global reference time support that meets the CPS challenges.

6. Load Balancing: Different load balancing aspects are needed in constrained resources CPS to enhance the utilization of CPS resources, to meet applications requirements, to enhance performance, and to increase the sustainability of the system. The load balancing aspects can be related to balancing distributed and parallel processing to enhance response time or to meet with time constraints, balancing the power consumption to enhance the sustainability of the system, and balancing the communication traffic to enhance throughput and data transfer times, in addition to balancing overall CPS operations.

7. Scalability support: some CPS applications involve a large number of components and extend over large physical environments or systems [73]. Examples of these applications are smart grids and gas and oil pipeline monitoring and control systems, where a large number of sensors and actuators are used covering extensive geographic areas. Designing such large-scale systems requires good middleware support to deal with large number of widely distributed components as well as high communication traffic generated from these CPS

components. In addition, some CPS applications could expand over time and include more components, services and sub systems. This growth could affect the overall performance of the system if not designed to scale well.

8. Supporting autonomous operations for complex CPS applications: Many CPS applications are considered complex systems incorporate many components that interact with each other for monitoring and controlling physical environments and systems. These complex systems can be in a huge number of different states at any point of time. It is generally extremely difficult to develop code to handle all these states effectively and in a timely manner. Having middleware that supports autonomous operations such as self-adaptive, self-resilient, and self-protected services [28] can relax implementing and operating these complex CPS applications. However, providing such services can be very challenging as it is not easy to predict all possible states and situations early in the design process. In addition, many of these self-x properties require complex algorithms and in some cases some intelligent components to be handled correctly. Moreover, testing verifying and validating these services when implemented is also challenging due to the large and complex set of possible combinations of events and states that could trigger them.

7. Current Research Efforts. There is some ongoing research to customize existing middleware platforms or design new middleware services to fit with the CPS challenges. One of these important challenges is supporting real-time operations in CPS. Real-time support requires provisions from the operating systems, resource managers and networks. The requirements and an architecture for a CPS middleware supporting these provisions was proposed in [31]. In addition, different real-time challenges for diverse scenarios were proposed. An example of these proposals is the approach to solve the real-time issue for aperiodic events in distributed CPS using a reconfigurable real-time middleware [82]. Another proposal is RDDS which is a publish/subscribe- middleware architecture developed to enable timely and reliable sensor data dissemination in highly unpredictable CPS environments [42]. There are also some research efforts dedicated to address the heterogeneity challenges in CPS. A middleware that provides interoperability between heterogeneous mobile devices in CPS was proposed in [76]. Furthermore, developing portable middleware services for heterogeneous CPS was proposed in [55]. Other research efforts were conducted to investigate reliability, security, safety, and fault tolerance in CPS. As an example, the main role of middleware in facilitating robust and resilient CPS was studied in [27] while a reliable, safe, and secure run-time platform for CPS was proposed in [51]. Moreover, a time-triggered middleware architecture that offers fault tolerance and dynamic reconfiguration at run-time taking into consideration the available system resources of the underlying infrastructure was proposed in [61].

Another group of research efforts were dedicated to investigating issues in large-scale CPS. The design, development, testing, and operations of a large-scale CPS are more complex compared to other CPS. This is due to higher heterogeneity, unreliability, unpredictability, complexity, and security requirements of large-scale CPS [46]. Therefore, large-scale CPS are very complicated to develop and operate without relying on support from advanced middleware services. Advanced middleware services can provide interoperability, reliability, QoS, and security mechanisms to satisfy the needs of large-scale CPS. In this regard, an efficient middleware for supporting distributed query processing in large-scale CPS was proposed in [26]. The work in [69] investigated developing a middleware on WSN for large-scale CPS. The aim of this middleware is to automatically achieve optimal sensor node configuration, bandwidth provisions, fault handling, and re-configuration in reaction to new missions and new added devices. In addition, a virtualized network platform for testbed of large-scale CPS was proposed in [16]. The requirements of virtual platform and networks for very large-scale CPS that expanded globally are investigated in [15].

A service-oriented approach to build middleware platforms for CPS was instigated into a number of research projects. This approach can solve many CPS challenges. A service-oriented middleware architecture to expose CPS devices to the Web was addressed in [37]. In addition, a service-oriented middleware for fog and cloud integrated CPS was proposed in [59] while a service-oriented approach to address fault tolerance in CPS was proposed in [21]. Another solution is a real-time service-oriented architecture middleware to monitor the performance and reserve resources in advance for CPS services in process to ensure its real-time achievability [52]. In the service-oriented middleware approach, system resources are viewed as a set of services to be used to develop CPS applications. One of the main advantages of this approach is the flexibility feature of extending the middleware itself to include new and more advanced services to support CPS applications as they develop. In addition, it provides the flexibility to add more devices, components, and services as the CPS grows or more

features are needed.

Some middleware platforms were also developed for specific applications or to solve specific issues in these applications. Examples include a middleware support for continuous monitoring of water distribution systems [40], a service-oriented middleware for smart grids [83, 54], a service-oriented middleware for collaborative UAVs, an event-driven middleware for smart buildings [63], an adaptive middleware for context-aware smart home applications [39], a middleware architectural framework for vehicular safety [72], an interoperable middleware platform for medical CPS [65]. Generally, these efforts provide solutions to specific issues in CPS rather than addressing the generic model that can support various features and apply to different CPS applications. More work is needed to address the general issues facing most, if not all, CPS applications and offer middleware platforms that can be adapted and used for several applications.

8. Open Issues. Based on the studied CPS applications and proposed middleware solutions, there are still a number of CPS middleware issues that need more research and deeper investigation leading to usable and effective solutions. The following is a discussion of some of these open issues:

1. Generic Middleware Architectures: There are several proposals for middleware architectures that are suitable of some CPS while they are unsuitable for others. These offer specialized solutions applicable to the application domain they target only. Developing a generic middleware architecture for all CPS applications is needed as they share several challenges that can be addressed effectively and reused for all CPS applications. The availability of such middleware architecture can provide a base for enhancing many solutions for many of the challenges in CPS. This also provides a common platform where features and services can be added, updated, enhanced or redesigned to benefit all applications. It can also enhance the development processes for CPS applications by offering more flexible design, implementation, testing, and reusability features. Moreover, using the same middleware platform for multiple CPS applications, will allow these applications to integrate easily and interoperate to achieve larger objectives.

2. Resource Management: Various efforts have been conducted to develop resource management techniques in traditional distributed systems. However due to the unique challenges of CPS, it is not easy to adapt these traditional techniques for CPS. Any adaptation of traditional resource management techniques into the context of CPS needs careful consideration of the efficiency, flexibility, and scalability of these techniques. Issues such as heterogeneity, varying capabilities, mobility, time constraints, and connectivity introduce more challenges for the resource manager. More investigation and proposals are needed for middleware-based resource management techniques in CPS.

3. Middleware Security Support: It is impossible to build secure and reliable CPS applications without considering the security and privacy aspects of these systems. Middleware can provide a number of security services for CPS. However, very limited research and development efforts were conducted in this regard. Meantime, a number of security middleware solutions were developed for other types of distributed systems such as ubiquitous applications [17]. These solutions provide a useful base for developing and operating ubiquitous applications. Similarly, security middleware solutions are needed for CPS.

4. Middleware Safety Support: As faults in CPS can cause severe and in many cases irreparable physical damages, middleware platforms should provide runtime environments for CPS applications that offer support for the safety of the CPS and its physical environment. More investigation is needed to find effective ways to utilize middleware for safety support. This can include, for example, developing middleware-based runtime validations and including fault detection and correction mechanisms within the middleware functions.

5. Middleware Sustainability Support: CPS are usually designed for critical applications that should live for a long period of time. However, some CPS devices have limited power, while others may have limited operational life. Over time many devices will need to be replaced for various reasons and the replacements may or may not be of the same type or capabilities. Designing any CPS solutions without considering the energy limitations, wear out probabilities and replacement devices will reduce the operational life span of the CPS applications. Careful designs are needed for all aspect of CPS to extend their life. This includes efficient energy aware features to conserve energy, unified interfaces to support device swaps and updates, and seamless integration of new devices and components. While this design can be extremely difficult, middleware platforms for CPS should assist in this regard. However, there are no comprehensive investigations toward developing middleware based solutions for sustainability support for CPS.

9. Conclusions. CPS applications are becoming an integral part of many environments and cover a diverse set of application domains. As a result, their design, development and operations have become complex and time consuming. The different types of applications have unique requirements and impose different challenges for the application developer. One method to help leverage some of the challenges and support the development process is to use middleware platforms. As discussed in this paper, middleware can provide various essential features and services for the CPS applications. In addition, it can also provide value-added features that enhance the operations and capabilities of the CPS applications. Unfortunately, such middleware, if designed to cover all needed aspects and functionalities of all types of CPS applications, will itself become too complex and difficult to design. We discussed the various challenges of middleware for CPS including the support for advanced communication schemes, effective and efficient resource management and load balancing, scalability, global time reference, security and autonomy. As discussed, many have addressed some of these challenges either on a generic basis for a single feature or specifically addressing one or a group of similar CPS applications. Unfortunately, there is a lot more to be done before a comprehensive middleware platform can be designed to support a large verity of CPS applications. A number of open issues need to be addressed and incorporated in a middleware approach. One of the main issues is the design of a generic middleware platform that can support different types of CPS applications. Other issues include the security and safety of the middleware in addition to resource management. Finally there is the issue of sustainability of the CPS application and its components. These issues have not been adequately addressed and require more efforts to create usable and efficient solutions. In our future work we intend to further investigate these issues and create possible approaches to address them. For example, we are looking into models for resource management for CPS environments that include stationary, mobile and limited resource devices.

REFERENCES

[15] S. Ahn, C. Yoo, S. Lee, H. Lee, and S. J. Kim, *Implementing virtual platform for global-scale cyber physical system networks*, International Journal of Communication Systems, 28 (2015), pp. 1899–1920.
[16] S. W. Ahn and C. Yoo, *Wip abstract: Virtual network platform for large scale cps testbed*, in Proceedings of the 2012 IEEE/ACM Third International Conference on Cyber-Physical Systems, IEEE Computer Society, 2012, p. 214.
[17] J. Al-Jaroodi, I. Jawhar, A. Al-Dhaheri, F. Al-Abdouli, and N. Mohamed, *Security middleware approaches and issues for ubiquitous applications*, in Computers and Mathematics with Applications, Special Issue on Advances in Cryptography, Security and Applications for Future Computer Science, Vol. 60, No. 2, 2010, pp. 187–197.
[18] J. Al-Jaroodi and N. Mohamed, *Middleware is still everywhere!!!*, Concurrency and Computation: Practice and Experience, 24 (2012), pp. 1919–1926.
[19] ———, *Service-oriented middleware: a survey*, Journal of Network and Computer Applications, 35 (2012), pp. 211–220.
[20] J. Al-Jaroodi, N. Mohamed, I. Jawhar, and S. Lazarova-Molnar, *Software engineering issues for cyber-physical systems*, in IEEE International Conference on Smart Computing (SMARTCOMP), 2016.
[21] P. Alho and J. Mattila, *Service-oriented approach to fault tolerance in cpss*, Journal of Systems and Software, 105 (2015), pp. 1–17.
[22] S. Ali, S. B. Qaisar, H. Saeed, M. F. Khan, M. Naeem, and A. Anpalagan, *Network challenges for cyber physical systems with tiny wireless devices: A case study on reliable pipeline condition monitoring*, Sensors, 15 (2015), pp. 7172–7205.
[23] P. Arjunan, N. Batra, H. Choi, A. Singh, P. Singh, and M. B. Srivastava, *Sensoract: a privacy and security aware federated middleware for building management*, in Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, pp, (2012), pp. 80–87.
[24] C. Berger and B. Rumpe, *Autonomous driving - 5 years after the urban challenge: The anticipatory vehicle as a cyber-physical system*, In Proceedings of the INFORMATIK, 2012 (2012), pp. 789–798.
[25] M. Chaqfeh and N. Mohamed, *Challenges in middleware solutions for the internet of things*, in proc, 2012 Int'l Conference on Collaboration Technologies and Systems (CTS), 2012.
[26] A. Cuzzocrea, J. Cecilio, and P. Furtado, *An effective and efficient middleware for supporting distributed query processing in large-scale cyber-physical systems*, in International Conference on Internet and Distributed Computing Systems, Springer International Publishing, ed., 2014, pp. 124–135.
[27] G. Denker, N. Dutt, S. Mehrotra, M. O. Stehr, C. Talcott, and N. Venkatasubramanian, *Resilient dependable cyber-physical systems: a middleware perspective*, Journal of Internet Services and Applications, 3 (2012), pp. 41–49.
[28] X. Dong, S. Hariri, L. Xue, H. Chen, M. Zhang, S. Pavuluri, and S. Rao, *Autonomia: an autonomic computing environment*, In the, 2003 (2003), pp. 61–68.
[29] P. T. Eugster, P. A. Felber, R. Guerraoui, and A. M. Kermarrec, *The many faces of publish/subscribe*, ACM computing surveys (CSUR), 35 (2003), pp. 114–131.
[30] Y. P. Fallah, C. Huang, R. Sengupta, and H. Krishnan, *Design of cooperative vehicle safety systems based on tight coupling of communication, computing and physical vehicle dynamics*, in Proc. of the 1st ACM/IEEE International Conference on Cyber-Physical Systems, ACM, 2010, pp. 159–167.

[31] M. Garca-Valls and R. Baldoni, *Adaptive middleware design for cps: Considerations on the os, resource managers, and the network run-time*, In Proceedings of the, 14 (2015).
[32] L. Gonda and C. E. Cugnasca, *A proposal of greenhouse control using wireless sensor networks. in computers in agriculture and natural resources*, American Society of Agricultural and Biological Engineers, 229 (2006).
[33] T. Gu, H. K. Pung, and D. Q. Zhang, *A service-oriented middleware for building contextaware services*, Journal of Network and Computer Applications, 28 (2005), pp. 1–18.
[34] R. Gummadi, O. Gnawali, and R. Govindan, *Macro-programming wireless sensor networks using kairos*, in International Conference on Distributed Computing in Sensor Systems, Springer Berlin Heidelberg, ed., 2005, pp. 126–140.
[35] V. Gunes, S. Peter, T. Givargis, and F. Vahid, *A survey on concepts, applications, and challenges in cyber-physical systems*, TIIS, 8 (2014), pp. 4242–4268.
[36] L. Gurgen, O. Gunalp, Y. Benazzouz, and M. Gallissot, *Self-aware cyber-physical systems and applications in smart buildings and cities*, in Proc. of the Conference on Design, Automation and Test in Europe, 2013, pp. 1149–1154.
[37] D. D. Hoang, H. Y. Paik, and C. K. Kim, *Service-oriented middleware architectures for cyber-physical systems*, International Journal of Computer Science and Network Security, 12 (2012), pp. 79–87.
[38] X. Hu, T. H. Chu, V. C. Leung, E. C. H. Ngai, P. Kruchten, and H. C. Chan, *A survey on mobile social networks: Applications, platforms, system architectures, and future research directions*, IEEE Communications Surveys & Tutorials, 17 (2015), pp. 1557–1581.
[39] M. C. Huebscher and J. A. McCann, *Adaptive middleware for context-aware applications in smart-homes*, In Proceedings of the, 2 (2004), pp. 111–116.
[40] M. Iqbal and H. B. Lim, *A cyber-physical middleware framework for continuous monitoring of water distribution systems*, in Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems, ACM, 2008, pp. 401–402.
[41] M. S. Kamal, S. Parvin, K. Saleem, H. Al-Hamadi, and A. Gawanmeh, *Efficient low cost supervisory system for internet of things enabled smart home*, in Conference on Communications Workshops (ICC Workshops),, Ieee International, ed., IEEE, 2017, pp. 864–869.
[42] W. Kang, K. Kapitanova, and S. H. Son, *Rdds: A real-time data distribution service for cyber-physical systems*, IEEE Transactions on Industrial Informatics, 8 (2012), pp. 393–405.
[43] S. Karnouskos, *Cyber-physical systems in the smartgrid*, in 9th IEEE International Conference on Industrial Informatics (INDIN, 2011, pp. 20–23.
[44] S. Kartakis, E. Abraham, and J. A. McCann, *Waterbox: A testbed for monitoring and controlling smart water networks*, in Proc, of the 1st ACM International Workshop on Cyber-Physical Systems for Smart Water Networks, 2015.
[45] A. Kesting, M. Treiber, and D. Helbing, *Connectivity statistics of store-and-forward intervehicle communication*, IEEE Transactions on Intelligent Transportation Systems, 11 (2010), pp. 172–181.
[46] W. T. Kim, I. G. Chun, S. H. Lee, H. Y. Lee, and J. M. Kim, *Wip abstract: From design to operation of a large-scale cps*, in Proceedings of the 2012 IEEE/ACM Third International Conference on Cyber-Physical Systems, Ieee Computer Society, ed., 2012.
[47] S. Lazarova-Molnar, H. R. Shaker, and N. Mohamed, *Reliability of cyber physical systems with focus on building management systems*, in proc. 2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC), 2016.
[48] E. A. Lee, *Cyber physical systems: Design challenges*, In Object Oriented Real-Time Distributed Computing (ISORC), pp, (2008), pp. 363–369.
[49] I. Lee, O. Sokolsky, S. Chen, J. Hatcliff, E. Jee, B. Kim, A. King, M. Mullen-Fortino, S. Park, and A. Roederer, *and k.k*, in Challenges and research directions in medical cyber–physical systems, In Proc. of the IEEE, 100(1, 2012, Venkatasubramanian, pp. 75–90.
[50] J. Lee, B. Bagheri, and H. A. Kao, *A cyber-physical systems architecture for industry 4.0-based manufacturing systems*, Manufacturing Letters, 3 (2015), pp. 18–23.
[51] S. S. Lim, E. J. Im, N. Dutt, K. W. Lee, I. Shin, C. G. Lee, and I. Lee, *A reliable, safe, and secure run-time platform for cyber physical systems*, in 2013 IEEE 6th International Conference on Service-Oriented Computing and Applications (SOCA), IEEE, 2013, pp. 268–274.
[52] K. J. Lin and M. Panahi, *A real-time service-oriented framework to support sustainable cyber-physical systems*, in 2010 8th IEEE International Conference on Industrial Informatics, IEEE, 2010, pp. 15–21.
[53] S. P. Lin and N. F. Maxemchuk, *The fail-safe operation of collaborative driving systems*, Journal of Intelligent Transportation Systems, 20 (2016), pp. 88–101.
[54] J. F. Martnez, J. Rodrguez-Molina, P. Castillejo, and R. De Diego, *R., "middleware architectures for the smart grid: survey and challenges in the foreseeable future,"* Energies, 6 (2013), pp. 3593–3621.
[55] K. Mechitov and G. Agha, *Building portable middleware services for heterogeneous cyber-physical systems*, In Proceedings of the Third International Workshop on Software Engineering for Sensor Network Applications, pp, (2012), pp. 31–36.
[56] L. Miclea and T. Sanislav, *About dependability in cyber-physical systems*, In, 2011 (2011), pp. 17–21.
[57] N. Mohamed, J. Al-Jaroodi, I. Jawhar, and S. Lazarova-Molnar, *A service-oriented middleware for building collaborative uavs*, Journal of Intelligent & Robotic Systems, 74 (2014), pp. 309–321.
[58] N. Mohamed, J. Al-Jaroodi, S. Lazarova-Molnar, and I. Jawhar, *Middleware to Support Cyber-Physical Systems*, in proc. IEEE Int'l Performance Computing and Communications Conference (IPCCC 2016), Las Vegas, Nevada, USA, 2016.
[59] N. Mohamed, S. Lazarova-Molnar, I. Jawhar, and J. Al-Jaroodi, *Towards service-oriented middleware for fog and cloud integrated cyber physical systems*, in IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), IEEE, 2017, pp. 67–74.

[60] M. M. Molla and S. I. Ahamed, *A survey of middleware for sensor network and challenges*, in ICPP 2006 Workshops, IEEE, 2006.
[61] A. Noguero, I. Calvo, and L. Almeida, *A time-triggered middleware architecture for ubiquitous cyber physical system applications*, in International Conference on Ubiquitous Computing and Ambient Intelligence, , Heidelberg, Berlin, 2012, Springer, pp. 73–80.
[62] L. Parolini, B. Sinopoli, B. H. Krogh, and Z. Wang, *A cyber–physical systems approach to data center modeling and control for energy efficiency*, in Proceedings of the IEEE, 100(1, 2012, pp. 254–268.
[63] E. Patti, A. Acquaviva, M. Jahn, F. Pramudianto, R. Tomasi, D. Rabourdin, J. Virgone, and E. Macii, *Event-driven user-centric middleware for energy-efficient buildings and public spaces*, IEEE Systems Journal, 10 (2016), pp. 1137–1146.
[64] T. Perumal, A. R. Ramli, C. Y. Leong, K. Samsudin, and S. Mansor, *Middleware for heterogeneous subsystems interoperability in intelligent buildings*, Automation in Construction, 19 (2010), pp. 160–168.
[65] J. Plourde, D. Arney, and J. M. Goldman, *Openice: An open, interoperable platform for medical cyber-physical systems*, in Conference on Cyber-Physical Systems (ICCPS),, Acm/ieee International, ed., IEEE, 2014, pp. 221–221.
[66] J. M. Portocarrero, F. C. Delicato, P. F. Pires, B. Costa, W. Li, W. Si, and A. Y. Zomaya, *Ramses: a new reference architecture for self-adaptive middleware in wireless sensor networks*, Ad Hoc Networks, 55 (2017), pp. 3–27.
[67] R. R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, *Cyber-physical systems: the next computing revolution*, in Proc. of the 47th Design Automation Conference, ACM, 2010, pp. 731–736.
[68] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, *Middleware for internet of things: a survey*, IEEE Internet of Things Journal, 3 (2016), pp. 70–95.
[69] N. Reijers, Y. C. Wang, C. S. Shih, J. Y. Hsu, and K. J. Lin, *Building intelligent middleware for large scale cps systems*, in 2011 IEEE International Conference on Service-Oriented Computing and Applications (SOCA), IEEE, 2011, pp. 1–4.
[70] K. Rmer, O. Kasten, and F. Mattern, *Middleware challenges for wireless sensor networks*, ACM SIGMOBILE Mobile Computing and Communications Review, 6 (2002), pp. 59–61.
[71] A. A. Salkham, R. Cunningham, A. Garg, and V. Cahill, *A collaborative reinforcement learning approach to urban traffic control optimization*, in Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, IEEE Computer Society, 2008, Volume 02, pp. 560–566.
[72] K. Saravanan, A. Thangavelu, and K. Rameshbabu, *A middleware architectural framework for vehicular safety over vanet (invanet)*, in First International Conference on Networks and Communications (NETCOM'09), IEEE, 2009, pp. 277–282.
[73] D. C. Schmidt, J. White, and C. D. Gill, *Elastic infrastructure to support computing clouds for large-scale cyber-physical systems*, in 2014 IEEE 17th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), IEEE, 2014, pp. 56–63.
[74] J. Shi, J. Wan, H. Yan, and H. Suo, *A survey of cyber-physical systems*, in 2011 International Conference on Wireless Communications and Signal Processing (WCSP), IEEE, 2011.
[75] S. Sridhar, A. Hahn, and M. Govindarasu, *Cyber–physical system security for the electric power grid*, in Proceedings of the IEEE, 100(1, 2012, pp. 210–224.
[76] J. Sun and Y. Zhang, *A middleware for highly dynamic distribution in cps environment*, in Proceedings of the 3rd International Conference on Context-Aware Systems and Applications, Social-Informatics and Telecommunications Engineering, 2014, ICST (Institute for Computer Sciences, pp. 169–172.
[77] Y. Tan, S. Goddard, and L. C. Perez, *A prototype architecture for cyber-physical systems*, ACM Sigbed Review, 5 (2008), p. 1.
[78] M. Tang, X. Dai, J. Liu, and J. Chen, *J., "towards a trust evaluation middleware for cloud service selection*, " Future Generation Computer Systems, 74 (2017), pp. 302–312.
[79] Q. Tang, S. K. S. Gupta, and G. Varsamopoulos, *Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach*, IEEE Transactions on Parallel and Distributed Systems, 19 (2008), pp. 1458–1472.
[80] S. M. Tonni, M. Z. Rahman, S. Parvin, and A. Gawanmeh, *Securing big data efficiently through microaggregation technique*, in IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), IEEE, 2017, pp. 125–130.
[81] Y. Yao and J. Gehrke, *The cougar approach to in-network query processing in sensor networks*, ACM Sigmod record, 31 (2002), pp. 9–18.
[82] Y. Zhang, C. Gill, and C. Lu, *Reconfigurable real-time middleware for distributed cyber-physical systems with aperiodic events*, in The 28th International Conference on Distributed Computing Systems (ICDCS'08), IEEE, 2008, pp. 581–588.
[83] L. Zhou and J. J. Rodrigues, *Service-oriented middleware for smart grid: Principle, infrastructure, and application*, IEEE Communications Magazine, 51 (2013), pp. 84–89.

Edited by: Amjad Gawanmeh
Received: May 30, 2017
Accepted: Nov 6, 2017

DOI 10.12694/scpe.v18i4.1333
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 18, Number 4, pp. 347–360. http://www.scpe.org

SMART SOLUTIONS FOR RFID BASED INVENTORY MANAGEMENT SYSTEMS: A SURVEY

ALI ALWADI[*], AMJAD GAWANMEH[¶], SAZIA PARVIN[§], AND JAMAL N. AL-KARAKI[♯]

Abstract. This article is a survey of the latest technologies, algorithms and state of the art localization techniques that can be used to serve as Internet of Things communication protocol by automating an RFID system. There is a lack of a reliable and up-to-date reference that can help inventory management systems developers and operators to enhance the management system efficiency, maximize the productivity, and minimize the material loss. Several low cost IoT devices and associated technologies, such as Radio Frequency Identification system, are widely used today in several applications, including educational, transportation, animal tracking, inventory object tracking, and so many others. In this paper, we present a survey of the state-of-the-art technologies, algorithms, and techniques used in smart Radio Frequency Identification systems based inventory systems. We first outline the design challenges for RFID-based inventory management systems followed by a comprehensive survey of various RFID technologies, RFID types, and RFID architectures. In addition, the latest researches in the RFID infrastructure and middle wares are evaluated. This includes passive RFID Tags, RFID Antennas, RFID middleware, and the RFID Reader. Finally, the paper presents the advantages and performance issues of different techniques in passive RFID, and investigates the collision and anti-collision algorithms for these types of applications.

Key words: Inventory Management System, Supply Chain, Smart Solutions, RFID, radio frequency identification, Internet of Things, IoT

AMS subject classifications. 90B05, 68M11

1. Introduction.

The recent civilization has forced a major expansion of cities around the world, which resulted in a continuous society and economic growth. This has driven the demand for mobility and automation of manual processes by users in order to make them smart, which will enhance the quality of life. Technology is being deployed in several infrastructures that are very complex and limited by its geographical location [46]. This resulted in several challenges in the design and implementation of devices to be used in such areas.

Mobilizing infrastructure requires a high level of balance between applying service solutions in order to meet the imperative mobility demands and to avoid a future mobility collapse. As a result, mobility related data must be gathered in order to be used to facilitate the public transport use, control traffic, connect emergency services with other health and the executive authorities, facilitate library processes and many other purposes which are too many to be enumerated in this paper [46, 34]. The engine that provides an environment to exchange mobility data between services is Internet of Things (IoT).

IoT can be simply defined in the scope of this article as a network that connects real-life objects into one network, using intelligent network devices, in order to exchange a specific set of information, to form an informational network that is integrated, based on standard communication protocols, Radio Frequency Identification (RFID) for instance. While there is no common definition for IoT, as it depends on the context of the application. In the context of this paper, it can be conceptually defined as a dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols. As physical and virtual "things" have identities, physical attributes, and virtual personalities and are seamlessly integrated into the information network. It can be seen as a new paradigm that includes a wide range of new technologies [4, 20, 17, 21, 3, 41].

RFID is the most common underlying communication protocol that serves the IoT concept, by connecting everyday objects using an RFID network. RF Identification is an object identification mechanism in which wireless communication technology is used as a communication protocol, which uses Radio Frequency (RF) signal for the communication between the RFID antenna and the tag. The physical objects are made active

[*]School of Engineering, Auckland University of Technology Auckland, New Zealand. ((ali.ab.alwadi@gmail.com)

[¶]Department of Electrical and Computer Engineering, Khalifa University, UAE. and Department of Electrical and Computer Engineering, Concordia University, Montreal, Canada. (amjad.gawanmeh@kustar.ac.ae).

[§]Melbourne Polytechnic, Victoria, Australia and University of New South Wales, Canberra, Australia. (saziaparvin@melbournepolytechnic.edu.au).

[♯]Department of Information Security Engineering Technology, Abu Dhabi Polytechnic, Abu Dhabi, UAE, and Dept. of Computer Engineering, The Hashemite University, Zarka 13115, Jordan. (jamal.alkaraki@adpoly.ac.ae).

network components by attaching a network interface to these objects. Using these interfaces, services will be able to interact with these "smart things/objects" that will provide the necessary link via the internet, to query and change their state, and retrieve any information associated with them, taking into security and privacy issues which will be solved by introducing the appropriate algorithms [5, 37, 3]. In our daily life, retrieving information from the connected things in the network will result in a huge data, which will require instant processing and analysis.

The revolution of cloud computing and IoT provides an opportunity to make technology touches unprecedented areas of our daily life. However, IoT consists of a number of heterogeneous devices and applications which their exchange information can differ in the format, size, and sometimes the protocol. That makes it hard for traditional cloud computing servers to recognize and process the exchanged information between the servers and the objects. Therefore, an improvement on the cloud computing servers is required to efficiently schedule and allocate IoT requests [36].

Cloud computing is a newly developed trend of the use of information technology (IT), which virtualizes the way computing resources are used and managed through new techniques. Cloud has shifted the IT industry to a new era with plenty of opportunities, induced a revolution in IT applications, and significantly influenced the way businesses use IT to create competitive advantaged. Commercially, the majority of the IT solutions providers recognize IT as a strategically crucial resource that contributed in the latest revolution, by bringing substantial benefits, such as cost saving, scalability, mobile storage, ease of access, better security, energy saving, and environment benefits [11, 44]. Cloud computing is a shared IT infrastructure where computing resources are scattered but linked together through the internet into a large pool of computing resources. That could automatically adjust the allocation of computing resources as the need for computing service fluctuates, resulting in higher utilization and productivity [11, 25].

Inventory management is a critical component of efficient supply chain management that is vital for success in a modern computing infrastructure. It is well-known that supply chain inventory management decisions depend on inventory data gathered from automated or manual control systems [49].

There are some complementary attributes between cloud computing and IoT. For instance, the centralized nature of cloud and the pervasive nature of IoT, the virtual nature of cloud supplements the real-world things in IoT, the ability to store a huge amount of information supplements the limited storage space for IoT [35].

In the next section, the concepts of IoT and Cloud will be briefly introduced, the relationship between them, and how IoT contributed in automating cloud based inventory management systems. In section 3, RFID Principles and Categories is explained. Section 4 explains possible middleware servers to run inventory management systems. And finally, conclusions and future work is explained in section 5.

2. Internet of Things and Cloud Systems in Inventory Management. Technological innovation and development has seemingly become the new de facto standard across many research facilities around the world. Technology users are wired in and logged on across the planet, each minute collectively generating or consuming 640 terabytes of data, photos, and videos on their handheld computing devices, tablets, and, more recently, wearable devices. In 2016, more than 5.5 million connected devices are added every day, and IoT is well on its way to involving more than 20.8 billion devices worldwide by 2020 [51].

2.1. IoT in Inventory Management. IoT is creating a new operation environment where each physical asset is individually identified with an ID, intelligently connected into a network, and digitally visible to the whole supply chain at a by-unit level and in real time [7]. IoT underlies almost all the emerging technologies used today. Mobile phones are the most prevalent example of popular technology. Driverless cars, is another application of this technology, which works through interconnectivity between the road, online maps, and other connected data, including weather predictions and traffic reports [51]. The BodyGuardian is a wearable sensor system that can remotely read a patient's biometrics (electrocardiogram, heart rate, respiration rate and activity level), sending data to the patient's physician and allowing users to go about their daily lives outside of a clinic facility [3, 51].

Similarly, by linking machines, products, people, and supply chain members, IoT provides a new environment for supply chain managers to manage and control inventory resources using applications that run on middleware servers [7]. The feasibility and flexibility of the architecture of this supply chain was proposed through a detailed implementation that uses wireless sensor networks and web services to introduce the use

of the smart object framework, these networks encapsulate RFID, sensor technologies, embedded object logic, object ad-hoc networking, and internet-based information infrastructure to realize the real-time monitoring of the flow of goods through a supply chain [54]. The implementation of RFID in the supply chain was employed to achieve real-time inventory monitoring and information sharing, such approach can help the system attain high environmental and economic benefits [54].

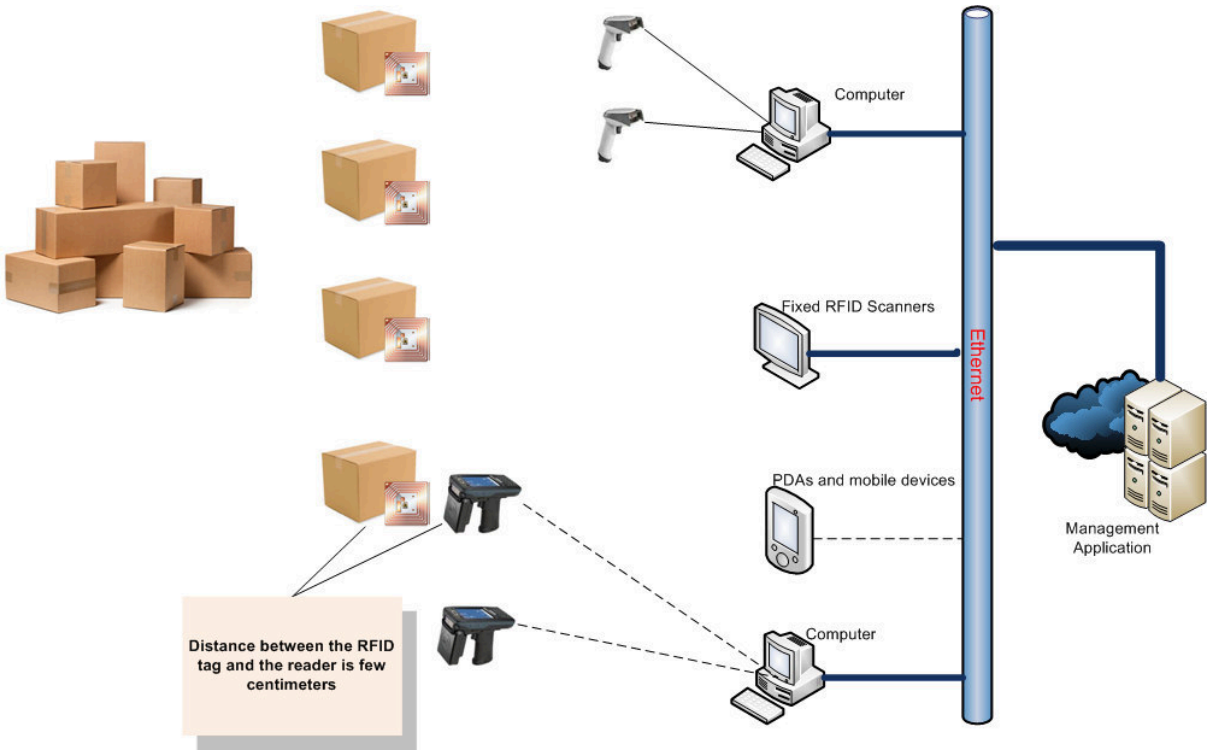


FIG. 2.1. *Inventory System Architecture*

The network of inventory objects is shown in Figure 2.1. By attaching an RFID tag to every item in the inventory, the tagged object becomes part of the network. The inventory administrator uses the handheld RFID readers to scan the objects, handheld readers nowadays can be wireless and smart with processing capabilities, they are connected to the middleware using any of the various computer cables or wireless using Wifi communication protocol IEEE 802.11. The reading distance between the reader and the tags cannot exceed a few centimetres, and both ends must be held appropriately to achieve the required polarization angle. The cloud is serving as the middleware provider for the management system.

RFID and associated technology extensive use of in several applications, including educational, transportation, animal tracking, inventory object tracking, and several industrial applications. This have made it recognized as a standard for IoT applications. Two main reasons contributed in this standardization, the increasing demand on this technology and the availability of RFID tags at very low prices. As a result, the cycle of manufacturing RFID-related products has speeded up lately [33].

2.2. Cloud Systems in Inventory Management. Cloud computing is a shared IT infrastructure where computing resources are scattered but linked together through the Internet into a large pool of computing resources and that could automatically adjust the allocation of computing resources as the need for computing service fluctuates [11, 15]. This results in a model for provisioning processes, applications and services that is potentially cost-efficient. In addition, it can make IT management simpler and increase business responsiveness as well [15].

The evolution of Cloud Computing concept has been driven by the accelerated shift in the computing

paradigm in the past half century. Initially, computing power existed mainly in mainframes shared by users using terminals. Standalone personal computers had their share of this revolution, they came powerful enough to process the users' tasks. This has encouraged another phase of this evolution. Users having powerful computers can now exchange information, and share processing power using computer networks. The next phase of the evolution is the internet, a network of local networks connected to form a more global network to utilise remote applications and resources. Finally, the concept of distributed systems came along, which is an computerized grid that facilitates sharing the computing power and storage across the network components. This forms the backbone of cloud computing, as cloud enables utilization of the available resources in a scalable and simple form [15], unlike a mainframe is a physical machine that offers finite computing power. Unlike mainframe, a cloud represents all possible resources on the Internet, offering infinite power and capacity.

Cloud computing applies distributed arithmetic techniques, which allows developers to more easily develop application services. It can automatically manage large numbers of computers for task and storage distribution. In 2003, the NSF invested USD 8.3 million to support the network virtualization and cloud computing VGrADS, was launched by the top U.S. seven colleges, which was when the research on cloud computing was born [14, 32].

Cloud plays a key role in RFID identification process through:

- Managing the hardware resources in the RFID network efficiently, RFID readers, antennas, and tags.
- Managing the information flow from/to the inventory repository regardless of the type of the database engine or the platform used.
- Running the inventory management software, and provide the users with a platform-independent, stable, and robust real-time management software. The user must interactively be able to access the management system, add, modify or delete inventory resources, and the changes must take effect instantaneously.

Cloud computing is organized into three standard service models [19], see Figure 2.2:

- Infrastructure as a Service (IaaS): The cloud vendor provides the servers (such as processing capability), storage (such as replication, backup, and archiving), and connectivity domains (such as firewalls and load balancing).
- Platform as a Service (PaaS): Describes a model in which the cloud vendor provides the platform that allows creation and deployment of applications and services the organization accesses through the Internet.
- Software as a Service (SaaS): The cloud vendor has complete control over the application, including capabilities, updates, and maintenance. The user is provided with a secure and versatile service accessible via the internet.

3. RFID Principles and Categories. RFID can be used to locate and track items in warehouses and during the entire shipping route as well [53]. Furthermore, RFID has extensive applications such as transportation and logistics, asset tracking, inventory management, and healthcare [58], object identification and tracking [56], security in airports, malls and military bases [31, 48, 53, 55, 56, 58], and so many others. In this article, an overview of the RFID technology and its history is first provided, then RFID types and architectures are explained. In addition, the article analyzes the latest developments of RFID infrastructure and middleware architectures, starting from passive RFID Tags, RFID Antennas, RFID middleware, and the RFID Reader. Finally collision, and anti-collision algorithms in RFID will be explained.

RFID networks can be divided into three main categories based on the tag types: Passive RFID, active RFID, and semi-passive RFID Tags. All three types contain integrated circuit that store tag information [33]. Passive tags are called passive because of the lack of power source, they depend on the energy transmitted from the reader antenna radio frequency signal to power up the tag. On the other hand, active tags contain a small battery that provides the integrated circuit with the power required to operate the tag circuitry, which is why active tags have better coverage, enhanced storage capacity, and they are often programmable. While passive tags on the other hand, have smaller read range, less memory capacity and they are usually programmed once [33, 40, 50].

Passive UHF RFID is being widely used in order to identify and track several types of inventory items as illustrated in Figure 3.1. Before moving to the technical aspects, one technical term that is tightly bound to RFID needs to be explained, which is Load Modulation. For the antenna to transfer data, a basic system of

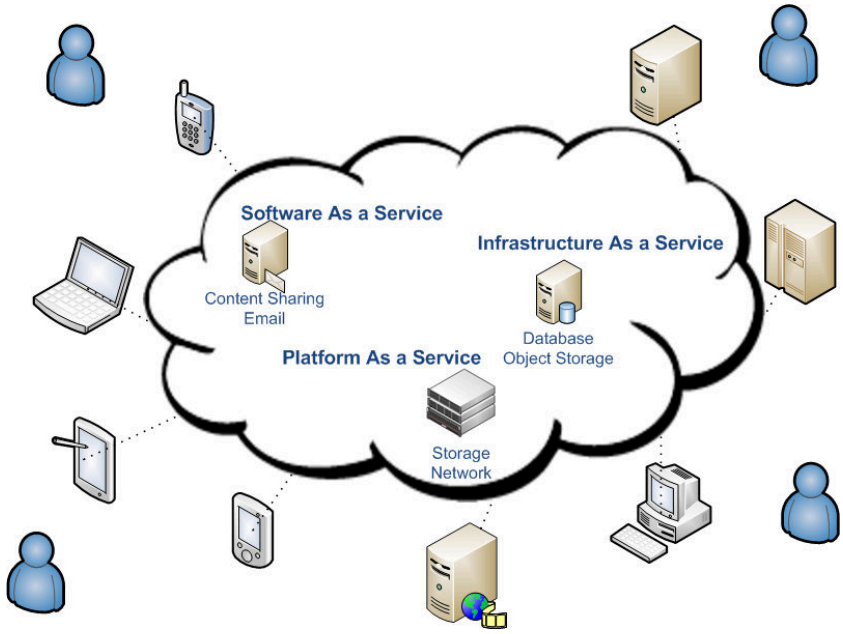


FIG. 2.2. *Inventory System Architecture*

modulation resistance, in parallel with the tag antenna is switched on and off at a rate governed by the signal data, the receiver on the other side will need to strip out the carrier signal [45]. The tag's antenna captures energy from the transmitted signal, and transfers the modulated signal which contains the tag ID. The receiver circuit is often called transceiver, it is responsible of managing the tag's power in general. In other words, it captures the energy received by the tag's antenna, and release it to feed the tag components. The receiver circuit has gained a considerable amount of research during the recent years [4, 10]. The memory chip contains a preprogrammed unique Identifier [4, 50].

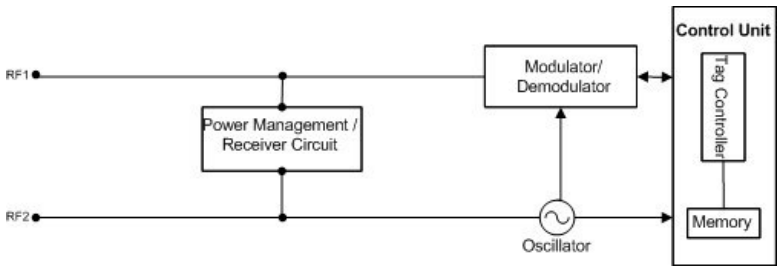


FIG. 3.1. *Passive RFID Tag Components [4]*

RFID tags are usually classified based on the ElectroMagnetic (EM) wave that is used to transfer the signal. The first category is Near-Field, which depends on the direct magnetic induction of the transmitted signal. The second type is Far-Field, which depends on capturing the energy release from the electromagnetic wave, which is a result of the transmitted radio wave from the antenna. The principle of the communication protocol in both techniques is similar. Both of them utilize the electromagnetic field attached to the RF signal to transfer information, the power transferred normally varies between 10 mW and 1 mW [50]. Far-field is generally used in applications where a long read range is needed. Near-field RFID is generally used to track tags in areas contain metallic barriers [18, 38, 4]. Inductive coupling technique is preferred in most near-field RFID applications mainly for one reason, the flexibility of using the load modulation techniques to transfer signals between the tag and the antenna and vice versa [39]. On the other hand, the electromagnetic field in far-field is radiative in

TABLE 3.1
Characteristics of Near-Field Vs. Far-Field

Property	Near-Field	Far-Field
Read Range	5mm - 10cm (Antenna Dependent)	Max. 22.1m
Modulation	Load Modulation using capacitive coupling	Electromagnetic radiation
Electromagnetic Signal	Radiative signal	Radiative signal
Reader Antenna	Small, Omnidirectional	Resonant, directional. Small antenna size/high frequencies
Usage	Metal or liquid surrounded objects	When long reading range is required

nature, making the choice of modulation techniques must narrower.

Coupling captures EM energy at a tag's antenna as a voltage disturbance. Part of the energy reaches a tag's antenna is reflected back due to an impedance mismatch between the antenna and the load circuit. The amount of reflected energy vary depending on the impedance of the antenna. This is called Backscattering [28]. The Table 3.1 clarifies the characteristics of Near-field versus Far-field and their usage.

3.1. Near Field RFID. Near-Field is a technology where magnetic coupling is used for communication between the reader and tag. In this technology, the modulated RF signal is transmitted through magnetic induction, where the reader passes an AC current through a coil inside the reader, which generates a corresponding alternating magnetic field. A capacitor is used to rectify this voltage, a reservoir of charge accumulates, which is enough to power the tag chip [33]. Figure 3.2 illustrates Near Field RFID technology.

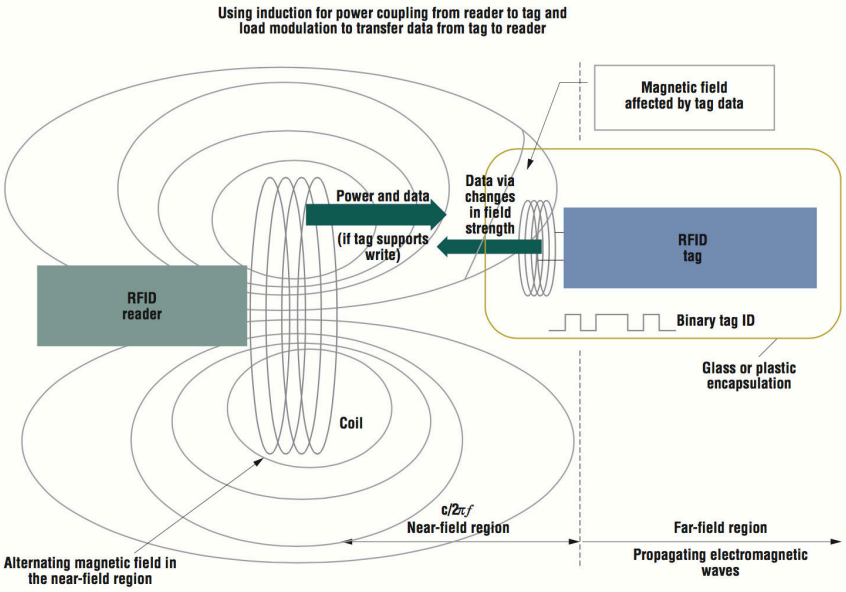


FIG. 3.2. *Load Modulation in NearField [50]*

Near-field coupling tags use low carrier frequencies. The two most commonly used are 128 kHz and 13.56 MHz. For instance, the maximum read distances are 372m for 128 kHz and 3.5m for 13.56 MHz. The disadvantages of using low frequencies in near-field are [4, 9]:

- To capture the RF signal, an antenna with a large coil is needed, as the power generated from the antenna is very low.
- The radiated power decreases heavily as distance increases.

- A low data rate resulted from the low bandwidth used.

Due to its design simplicity, Near-Field system was one of the first RFID techniques to be used for the implementation of RFID tags, in particular, passive ones. This technology, however, suffers from physical limitations due to the need of very close contact between the tag and the antenna in order to produce coupling. The range of magnetic induction that can be used is highly dependent on the tag's frequency. As the frequency of operation increases, the distance over which near-field coupling can operate decreases. The energy available for induction is another limitation as a function of distance from the reader coil. As distance increases, the magnetic field drops off at a factor, which can be calculated in the equation [33, 58]:

$$F = \frac{1}{r^3} \tag{3.1}$$

F is the drop factor, and r is the distance between the antenna and the tag. Since applications require maintaining a fixed read rate, tags with with various polarization poles must be detectable, as a result, more tag requirements have arouse, for instance higher read rate and higher radio frequency. These requirements have resulted in using far-field in passive RFID identification [33].

Power path transmission loss in near field is defined in literature as:

$$P_r = K\frac{1}{r^\alpha}P_t \tag{3.2}$$

where P_r is the received power, K is constant that depends on antenna gain and wavelength, P_t is the transmitted power. In addition, for near field α can be considered as: $2 < \alpha < 4$.

The read range for near-field coupling is narrow.The conventional solid-line loop antennas with one operating wavelength can produce uneven magnetic field over the interrogation area, since phase-inversion occurs across the loop conduction material, and therefore the electrical current reaches zero as it passes the antenna material. The reliability of RFID tag detection depends on the strength of the magnetic field. Hence, the relatively weak field in far end regions of the interrogation zone results in degradation of the detection reliability. As a result, the reader antenna must be redesigned to accommodate a larger antenna in order to maximize the coverage area. For instance, a tag antenna of size $150 \times 150\ mm^2$ can compensate the shortage in the read distance. [33, 58].

3.2. Far Field RFID. To transfer signals, Far-field system utilizes the electromagnetic field that results from the emittance of the radio frequency signal. Far-field systems operate at the UHF region of the spectrum, normally between 840 MHz and 960 MHz, or microwave frequencies, between 2.45 GHz and 24 GHz [8]. A dipole antenna that is imbedded in the reader antenna transmits EM waves to the tag. The EM wave triggers an alternating potential difference once received by the tag's small dipole antenna, which appears across the antenna circuit, this forms an alternative current that is passed through to a diode which is also linked to a capacitor, which will then be rectified as an energy that the tag uses to power up the chip. However, unlike inductive coupling, tags are beyond the range of the reader's EM wave area, and information cannot be transmitted back to the reader using load modulation [50].

A far-field system's range has some limitations, which can be summarized:

1. The amount of energy received by the tag.
2. The sensitivity of the reader's antenna to the signal received from the tag.

As the electromagnetic wage travels, attenuation occurs on both sides of communication, between the antenna and the tag, and the response signal from the tag. The received signal is expected to be weak. The returning energy from the tag is [33, 58]:

$$F = \frac{1}{r^4} \tag{3.3}$$

where r is the distance.

Nowadays, the semiconductor revolution has helped in reducing the size of the electronic materials in general, which helped in decreasing the power necessary for these components to consume in order to operate as expected, RFID tag has had its share of this innovation, power consumption continues to decrease, which helped

in expanding the RFID operations and usages. In addition, low cost customized RFID tags can be designed and manufactured. These can efficiently read from a wider distance which sometimes can reach up to 6 meters with power usage as low as 100 dBm [50].

3.3. RFID based Communications. RFID systems operate in a different range of the spectrum depending on the antenna and tag design, which can vary from 100 kHz to 5.8 GHz. The underlying technology requires the RFID systems to operate in the UHF frequencies occupying the ISM bands 860 - 960 MHz, according to frequency band allocation in each country. The read range offered by UHF RFID makes this frequency the most attractive for use by supply chain management. The allocated band at ultra-high frequency (UHF) ranges from 860 MHz to 5.8 GHz [33, 16, 13].

3.4. RFID Tag. RFID tags are passive, active or semi-passive. The term, passive or active is referred to the power source that the tag uses to extract the required energy to operate. Passive tags retrieve the energy from the electromagnetic signal transmitted from the reader's antenna. The tag retrieves the required energy from the interrogating wave or from a modern antennas have a dedicated port that transmit constant signal to power up tags in the range. The second type of tags is active tag, which have internal batteries as a power source, to enhance the reading range. Passive tags have the lowest price in terms of manufacturing cost, they do not require maintenance, more compact and lighter [33, 16].

Semi-passive tags are similar to the passive tags in the way they operate, they use backscattering technique to respond to the reader. The main difference is that semi-passive tags have batteries to power the chips that are embedded in the tag, which is used in conjunction with integrated electronic components such as sensors. Semi-passive tags have almost the same characteristics of the passive tag (reading range, operating frequency) [13].

RFID tags contain RF transponders with digital memory chips that are uniquely identified. The antenna packaged with a transceiver and decoder, emits a signal activating the tag, refer to the Figure 3.3 [4, 33]. A brief description of each RFID tag component is mentioned below [52]:

1. RF Interface The radioactive component of the tag, which performs the following:
   - Supply RFID transponders with power by generating the required energy.
   - Modulating the signal in preparation for the transponder to transmit.
   - Reception and demodulation of signals received by the transponders.
2. Control Unit As the name implies, this part of the tag controls the tag operation, by performing the following functionalities:
   - Execution of the application software commands.
   - Signal interpretation.
   - Controls the communication with the transponder.
   - Anti-collision, and encryption can be implemented in the control unit.
   - Transponder-reader authentication.

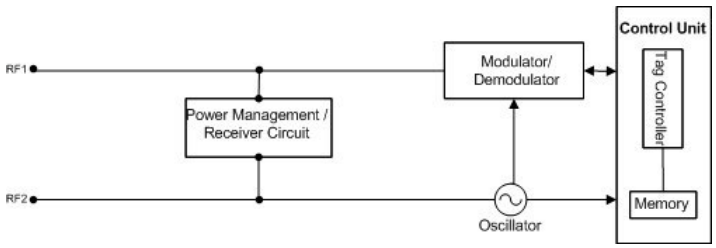


FIG. 3.3. *Block Diagram for Passive RFID Structure*

The capacity of the tag varies depending on the specifications of the vendor. It is very common ti find most modern tags with a capacity of up to 2,048 bits of information [33, 52]. This is enough to store the necessary information required about library items.

3.5. RFID Reader Antenna. As explained earlier, RFID reader antenna transmits signals as electromagnetic wave called radio frequency wave. Antennas can be classified into two main types based on the characteristics of the transmitted signal [26]:

- **Linear Polarization**: The electromagnetic wave propagates in a single direction, vertically or horizontally, depending on the orientation. These antennas are also referred to as Dipole antennas. This type of antennas is recommended to use when the tag has fixed orientation. Therefore, it is commonly used in hand-held readers in warehouses and inventories. In order for this antenna to work, the reader and the tag must be aligned together. This is a major drawback of this type of tags.
- **Circular Polarisation**: The electromagnetic wave covers the two planes during propagation, in a circle-like motion that resembles the motion of a screw, which makes this type more efficient than the linear polarization for two main reasons: The wide coverage area, and the tag orientation does not have to be inline with the antenna in order to make a successful read. One disadvantage of this antenna is energy loss, since the majority of the energy in the transmitted signal lies within the first few waves, as illustrated in Figure 3.3.

RFID antennas have also a commercial classification that divides antennas into two main groups, based on the number of ports that the antennas have [26]:

- **Monostatic Circular**: This antenna is equipped with a single port, which is used for transmission and receiving of the RFID signals.
- **Bistatic Circular**: This antenna has a dual ports, one for transmission and one for receiving signals. This type of antennas is more common, but it is more expensive.

Both antennas can be manufactured with a Listen Before Talk (LBT) port, which is a dedicated port that listens for signals before sending RFID signals. The read range for both types depends on [29]:

- The available power at the reader/interrogator to communicate with the tags.
- The quality of the tag's circuitry, and its ability to capture the released energy to power the tag's components.
- The environmental conditions and structures, as the operation frequency increases, the probability of the signal being obstructed by metal or a wall is higher.

The released wave from an antenna propagates in space in a circular motion, its strength diminishes as the traveled distance increases. The design of the antenna determines the shape of the delivered wave, as a result the read range and the positive identification of tags are also affected by the distance between the reader and the tag, as well as the orientation of the antenna. In an ideal situation, free of obstructions or absorption objects, the signal strength decay in inverse proportion to the cube of the distance (refer to equation 2) [22]. When an antenna transmits a power signal, regardless if it was a constant power signal or a regular RF signal, the tags in the range are powered up and will reply with their identification signal, for a large number of tags, this will cause an issue, it is called collision, which heavily affects the efficiency of identification.

3.6. Collision and Anti-Collision Algorithms. Collision is an undesired interference that occurs between radio signals initiated from two or more radio frequency components that lie within the frequency range of each other. Normally, an inventory system equipped with passive RFID contains a high density of RFID equipment that transmit signals around, collision happens often when two or more radio active components reply to the reader simultaneously. Collision is a critical terminology in RFID networks for the following reason, when the reader antenna transmits the power up signal, which normally happens by sending the read signal, all tags within the reach of reader signal will detect and respond to it with their modulated signals. Collision occurs here as a result of the antenna receiving tens and sometimes hundreds of response signals from the tags. An algorithm is required to "detect" or organize each tag's signal and marshal the signals in orderly manner without interference of signals from other antennas or tags, applying anti-collision algorithms enhances the network throughput by reducing the time required to perform the identification. Various anti-collision algorithms have been proposed, the main two algorithms: (a) ALOHA-based algorithms, the frequency range is divided into time slots, each time slot is assigned to one tag only. (b) Tree-based algorithms that group the tags into a tree with branches that contain subsets. The identification of tags is achieved by iterating through the tree branches [4].

ALOHA is an algorithm built based on reducing the probability of tag collision by assigning time slots to

tags. It utilizes Time Division Multiple Access (TDMA) to implement a collision resolution [30]. This document focuses on ALOHA-based algorithms for two main reasons, they work well with Read-Only tags, in other words, they don't require the tag to be reprogrammed to hold extra information. The second reason is simply because they are the most frequently used anti-collision techniques. Dynamic Frame slotted Aloha (DFS-ALOHA) in particular will be discussed in this article. By anticipating the probability of collision to estimate the number of tags in the interrogation zone, the algorithm determines the appropriate frame size depending on the number of tags. To avoid collision, DFS-ALOHA algorithm allocates a random frame for each tag to transmit data in. The bandwidth is divided into frames, and the frames are divided into slots. The system efficiency depends heavily on numerous factors: The amount of tags, the distribution of tags around the antenna, and the estimated frame size [23, 47, 57]. However, ALOHA-based algorithms have one disadvantage, "Starvation". A tag may not be allocated a time slot to use to load its signal, which makes it undetectable by the surrounding antenna(s), as a result, the tag may not be detected for a a period of time, and sometimes forever [12].

Tree-based algorithms are deterministic, they use a search tree to identify tags that fall in the frequency range of the antenna by classifying the tags in the interrogation zone into a tree. The algorithm iterates through the branches until all tags have been identified. Tree-based algorithms are divided into three types: Binary Tree (BT), Query Tree (QT) , and finally Binary and Query Tree, which is a combination of both [16, 24, 27]. QT uses tag IDs to group the tags into subsets, then the system iterates through the subsets by dividing all subsets into groups that contain a set of two. The efficiency of the identification process is affected by the distribution of tags in the system. BT uses randomly generated numbers to identify the tag groups, which requires reprogrammable tag memory to store the assigned number in order to make this algorithm more efficient, however, in case of passive RFID reprogramming the tag chips are not programmable [6].

4. Middleware for Inventory Management. The Middleware is a software that facilitates the exchange of information between the applications, repositories and the integrated hardware. It is an engine that deploys the binaries and application executable files which are responsible of managing the flow of data between the antennas, readers and the management application. The middleware application server operates the integrated components, and manages the flow of information between the application and the peripherals. It also supports readers with the connectivity, context filtering and message routing, and integration with the server interfaces. To achieve successful tag identification, the middleware must achieve the following [6]:

1. Real-time processing of transaction events from the hardware components.
2. The middleware must provide a common interface to access different kinds of hardware offering different features.

RFID middleware is composed of four main layers [6]:

- Reader Interface: This layer is close to the RFID hardware. It manages the flow of information with the RFID hardware components. It manages the hardware interface related parameters, for instance reader protocol, and reader interface.
- Data Processing and Storage: Processing the data received from the readers, and storing the transformed information into a repository.
- Application Interface: Providing the user with user friendly interface required to control the RFID Middleware application and configure the components attached to it.

4.1. IBM WebSphere Premises Server. WebSphere Premises Server is a framework that is based on the foundation of a service-oriented architecture (SOA). It can efficiently deliver several services though which sensor integration solutions are supported. As a result, this server can provide a robust, flexible, and scalable platform for capturing application-driven information from sensor data. This platform can be used for integrating new sensor data, identifying the relevant application events using situational event processing, and then integrating and acting upon those events with the deployed SOA application processes, which are designed to handle those events, process them, and store them in the designated repository [15]. The middleware application must be built with an integrated sensor-based functionality in order to retrieve data from sensors, and build business transactions out of these raw input data that can lead into business decisions. WebSphere Premises Server delivers the platform of scalable, reliable end-to-end sensor business solutions by [15]:

- Extending IBM SOA process integration platform to integrate applications with sensors to provide them with sensor data and events allowing system administrators to flexibly deploy applications at runtime

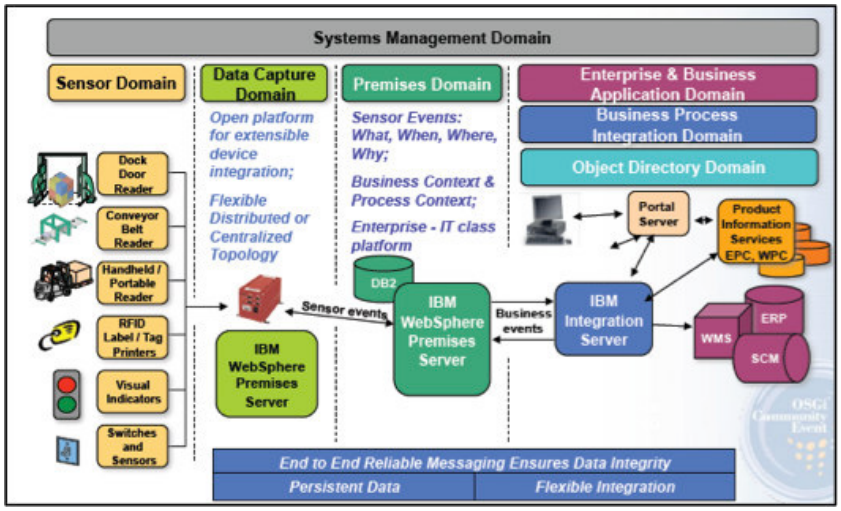


FIG. 4.1. *WebSphere Middleware Architecture [15]*

without the need to reorganize or restart the server.

- Providing an engine with intelligent business rules to develop identification for complex events from multiple types of sensor data.
- The server is bundled with system features to support real-time location tracking services, including device interfaces for RFID components, core localization processes and a graphical user interface for location visualization.
- Enhancing device driver services, allowing a single core platform to support multiple sensor types including passive RFID, active RFID and sensors.
- Providing an Integrated Development Environment (IDE) friendly platform for integrating sensor devices using customized applications.
- Integration with WebSphere RFID Information Centre to enable the deployed applications to efficiently manage and integrate sensor information with enterprise applications as well as securely share sensor information and events with selected third party applications.
- WebSphere Server stack contains three main components as shown in Figure 4.1: RFID devices, WebSphere Premises Server, and WebSphere Business Integration Server [15].

4.2. Rifidi Edge Server. Rifidi Edge Server is an engine that connects the IoT components with people who use handheld and mobile devices, over the cloud. It is a complete RFID Middleware Platform with built-in integrated development tools to enable the development and deployment of customized RFID applications. The product is an open source alternative to popular RFID platforms such as IBM Premises Server and Microsoft Biztalk RFID. Built on the latest Java OSGI platform and integrated with powerful open source rules engine (Esper). Rifidi Edge has the ability to build complex applications that interact with the most popular RFID and sensor devices available in the market [43].

The most fundamental functionality for Rifidi Edge Server is to gather data from sensors, and deliver them to middleware applications that use the data for processing and storing [42]. The server filters out all the noises and distorted signals that the sensors deliver to the middleware, which is an important factor in the RFID area in order to filter out all the undesired radio signals.

Figure 4.2 contains a high-level description of how data are collected and transferred through the edge server. Sensor and Reader Abstraction layer is the first module that received data from the sensors or any connected RFID component, which normally contains customized programming interfaces to interact with the custom sensors. While these components are normally hardware RFID readers, such as Alien 9800 and Symbol XR400. Data might also be produced by a legacy barcode reader, a database, or even another edge server.

As data are collected from the sensors, they are passed into a high-speed internal message queue system in

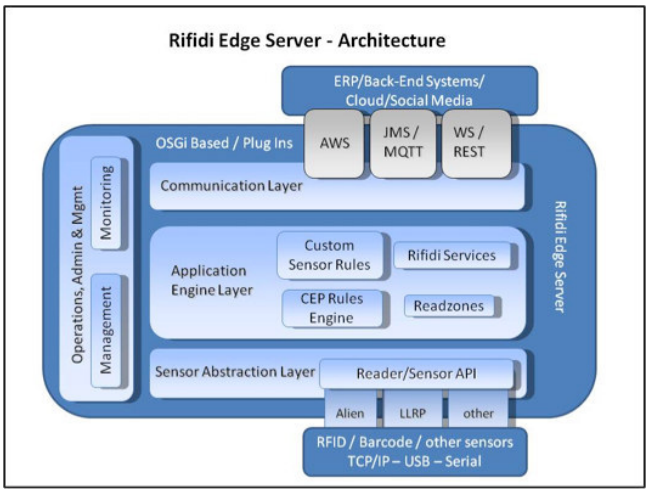


FIG. 4.2. *Edge Server data flow*

order to be exchanged with other server components. The Application Layer Events (ALE) provides the system with an interface to collect and filter RFID data. Data collection according to ALE rules is achieved by using an event stream processor called Esper [42, 29].

5. Conclusions and Future Work. This article is intended as a survey for the latest available research in Cloud Computing, inventory management middleware servers, and RFID Identification and localization algorithms that can be used to automate an inventory that has a passive RFID infrastructure. As illustrated in Table 5.1, several technologies that can be potential candidates to be used to automate an inventory system using cloud are summarized in this work. This article is a baseline for the system owners and implementers to locate inventory objects (or the Tagged objects) at runtime, without having to replace (or possibly reprogram) the passive RFID tags. Using the proposed algorithms, technologies allow the system owner to achieve the following:

- Automate the object localization, and remotely control the automation process using simple but efficient

TABLE 5.1
Summary of technologies used in smart inventory systems

Group	Reference	Technology Used	Problem Addressed
IoT	[34] [46]	IoT devices	Utilizing IoT Concepts in mobilizing cities
Hybrid	[3] [4] [5] [17] [20] [21] [31] [33] [37] [48] [53] [54] [55] [56] [58]	IoT and RFID	Using RFID as a communication protocol for IoT
Cloud	[11] [15] [19] [36] [44] [51]	Cloud Computing and IoT	Cloud Computing serving IoT applications
Inventory	[7] [14] [32] [35] [49]	Inventory Management	Automating an Inventory Management system using IoT concept
RFID	[4] [8] [10] [13] [16] [33] [45] [50] [52] [58]	RFID and Passive RFID Tag infrastructure	Available RFID types, and the tag structure and operating frequency of each type
Antenna	[22] [26]	RFID Reader Antenna	RFID Antennas and wave types
Middleware	[6] [15] [29] [42] [43]	RFID Middleware	Middleware for Inventory Management applications

management system, which can be accessed anywhere, anytime.
- Dynamically add or remove system resources, RFID objects from the network with having the need to modify the management or the tracking code.
- Implement a fault-tolerant system, which can detect hardware and software failures

Based on the performed survey, there are several issues that can be addressed in order to provide reliable RFID based inventory system, such as automated library system. First, a hardware prototype as well as a simulation software are required, in order to capture all design specifications, including operating frequency, read range, noise factor, energy loss. The simulation results must be recorded and matched against the hardware implementation. Next, a chosen middleware need to be configured and integrated with the workbench in order to start the hardware integration with the RFID readers. New Command line interface must be developed to integrate the middleware server with the reader antennas. Initial simulation results proved that such system will be efficient and reliable for the purpose of automating inventories.

REFERENCES

[3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376, Fourthquarter 2015.

[4] Ali Alwadi, Jeff Kilby, Amjad Gawanmeh, and Hakilo Sabit. Radio frequency identification technology in libraries. *International Journal of Engineering Research and Science*, 2(11):65–77, November 2016.

[5] R. E. A. Anee and N. C. Karmakar. Chipless rfid tag localization. *IEEE Transactions on Microwave Theory and Techniques*, 61(11):4008–4017, Nov 2013.

[6] R. Bhattacharyya, C. Floerkemeier, and S. Sarma. Low-cost, ubiquitous rfid-tag-antenna-based sensing. *Proceedings of the IEEE*, 98(9):1593–1600, Sept 2010.

[7] Li Bo and Li Yulong. Internet of things drives supply chain innovation: A research framework. *International Journal of Organizational Innovation*, 9(3):71 – 92, 2017.

[8] D. Caldwell-Stone. Rfid in libraries. *Library technology reports*, 46(8):38–44, 2010.

[9] V. Chawla and D. S. Ha. An overview of passive rfid. *IEEE Communications Magazine*, 45(9):11–17, September 2007.

[10] Jiann-Liang Chen, Ming-Chiao Chen, Chien-Wu Chen, and Yao-Chung Chang. Architecture design and performance evaluation of {RFID} object tracking systems. *Computer Communications*, 30(9):2070 – 2086, 2007.

[11] Thomas Chen, Chuang Ta-Tao, and Nakatani Kazuo. The perceived business benefit of cloud computing: An exploratory study. *Journal of International Technology and Information Management*, 25(4):101 – 121, 2016.

[12] W. T. Chen. An accurate tag estimate method for improving the performance of an rfid anticollision algorithm based on dynamic frame length aloha. *IEEE Transactions on Automation Science and Engineering*, 6(1):9–15, Jan 2009.

[13] Xun Chen and Z. Jane Wang. Reliable indoor location sensing technique using active rfid. In *2010 The 2nd International Conference on Industrial Mechatronics and Automation*, volume 1, pages 160–163, May 2010.

[14] O. Chieochan, A. SaoKaew, and E. Boonchieng. An integrated system of applying the use of internet of things, rfid and cloud computing: A case study of logistic management of electricity generation authority of thailand (egat) mae mao lignite coal mining, lampang, thailand. In *2017 9th International Conference on Knowledge and Smart Technology (KST)*, pages 156–161, Feb 2017.

[15] IBM Corporation. Ibm perspective on cloud computing, https://www-935.ibm.com/services/in/cio/pdf/ibm_perspective_on_cloud_computing.pdf, November 2016.

[16] Mehdia Ajana El Khaddar, Mohammed Boulmalf, Hamid Harroud, and Mohammed Elkoutbi. Rfid middleware design and architecture. In *Designing and Deploying RFID Applications*. InTech, 2011.

[17] P. Friess. *Internet of Things-Global Technological and Societal Trends From Smart Environments and Spaces to Green ICT*. River Publishers, 2011.

[18] F. Fuschini, C. Piersanti, L. Sydanheimo, L. Ukkonen, and G. Falciasecca. Electromagnetic analyses of near field uhf rfid systems. *IEEE Transactions on Antennas and Propagation*, 58(5):1759–1770, May 2010.

[19] G. Garrison, S. Kim, and R. L. Wakefield. Success factors for deploying cloud computing. *Communications of the ACM*, 55(9):62 – 68, 2012.

[20] D. Giusto, Antonio Iera, Giacomo Morabito, and Luigi Atzori. *The Internet of Things: 20th Tyrrhenian Workshop on Digital Communications*. Springer Science and Business Media, 2010.

[21] A. Gluhak, Srdjan Krco, Michele Nati, Dennis Pfisterer, Nathalie Mitton, and Tahiry Razafindralambo. A survey on facilities for experimental internet of things research. *IEEE Communications Magazine*, 49(11), 2011.

[22] Y. He and X. Wang. An aloha-based improved anti-collision algorithm for rfid systems. *IEEE Wireless Communications*, 20(5):152–158, October 2013.

[23] X. Huang and S. Le. Efficient dynamic framed slotted aloha for rfid passive tags. In *The 9th International Conference on Advanced Communication Technology*, volume 1, pages 94–97, Feb 2007.

[24] Xu Huang and Son Le. Efficient dynamic framed slotted aloha for rfid passive tags. In *Advanced Communication Technology, the 9th International Conference on*, volume 1, pages 94–97. IEEE, 2007.

[25] IBM. Ibm perspective on cloud computing, 2011.

[26] Skyrfid Inc. Reader antenna tutorial - what you need to know. http://skyrfid.com/rfid_antenna_tutorial.php, 2015.
[27] Y. C. Lai and L. Y. Hsiao. General binary tree protocol for coping with the capture effect in rfid tag identification. *IEEE Communications Letters*, 14(3):208–210, March 2010.
[28] J. Landt. The history of rfid. *IEEE Potentials*, 24(4):8–11, Oct 2005.
[29] Marc Langheinrich. A survey of rfid privacy approaches. *Personal and Ubiquitous Computing*, 13(6):413–421, 2009.
[30] Shian Liu and Xiaojuan Peng. Improved dynamic frame slotted aloha algorithm for anti-collision in rfid systems. *Knowledge Discovery and Data Mining*, pages 423–430, 2012.
[31] L. Lu, Y. Liu, and X. Y. Li. Refresh: Weak privacy model for rfid systems. In *2010 Proceedings IEEE INFOCOM*, pages 1–9, March 2010.
[32] Yi-Wei Ma, Wei-Ting Cho, Jiann-Liang Chen, Yueh-Min Huang, and Rongbo Zhu. Rfid-based mobility for seamless personal communication system in cloud computing. *Telecommunication Systems*, 58(3):233–241, 2015.
[33] C. R Medeiros, Jorge R Costa, and Carlos A Fernandes. Rfid reader antennas for tag detection in self-confined volumes at uhf. *IEEE Antennas and Propagation Magazine*, 53(2):39–50, 2011.
[34] A. Monzon. Smart cities concept and challenges: Bases for the assessment of smart city projects. *Communications in Computer and Information Science*, 579:17–31, 2015. cited By 0.
[35] G. Muhammad, S. M. M. Rahman, A. Alelaiwi, and A. Alamri. Smart health solution integrating iot and cloud: A case study of voice pathology monitoring. *IEEE Communications Magazine*, 55(1):69–73, January 2017.
[36] Husnu S. Narman, Md. Shohrab Hossain, Mohammed Atiquzzaman, and Haiying Shen. Scheduling internet of things applications in cloud computing. *Annals of Telecommunications*, 72(1):79–93, 2017.
[37] S. Preradovic and N. C. Karmakar. Chipless rfid: Bar code of the future. *IEEE Microwave Magazine*, 11(7):87–97, Dec 2010.
[38] X. Qing, Z. N. Chen, and C. K. Goh. A uhf near-field/far-field rfid metamaterial-inspired loop antenna. In *Proceedings of the 2012 IEEE International Symposium on Antennas and Propagation*, pages 1–2, July 2012.
[39] X. Qing, C. K. Goh, and Z. N. Chen. A broadband uhf near-field rfid antenna. *IEEE Transactions on Antennas and Propagation*, 58(12):3829–3838, Dec 2010.
[40] Bo Rundh. Radio frequency identification (rfid) invaluable technology or a new obstacle in the marketing process? *Marketing intelligence & planning*, 26(1):97–114, 2008.
[41] A. Sajid, Haider Abbas, and Kashif Saleem. Cloud-assisted iot-based scada systems security: A review of the state of the art and future challenges. *IEEE Access*, 4:1375–1384, 2016.
[42] Rifidi Edge Server. Rifidi edge server user's guide.
[43] Transcends-Rifidi Edge Server. Rifidi edge server, http://transcends.co/rifidi-edge-server/, 2016.
[44] Ayob Sether. Cloud computing benefits. *Browser Download This Paper*, 2016.
[45] S. Shao and R. J. Burkholder. Item-level rfid tag location sensing utilizing reader antenna spatial diversity. *IEEE Sensors Journal*, 13(10):3767–3774, Oct 2013.
[46] Michael Strasser and Sahin Albayrak. A pattern based feasibility study of cloud computing for smart mobility solutions. In *Resilient Networks Design and Modeling (RNDM), 2016 8th International Workshop on*, pages 295–301. IEEE, 2016.
[47] W. Su, N. Alchazidis, and T. T. Ha. Multiple rfid tags access algorithm. *IEEE Transactions on Mobile Computing*, 9(2):174–187, Feb 2010.
[48] C. C. Tan, B. Sheng, and Q. Li. Secure and serverless rfid authentication and search protocols. *IEEE Transactions on Wireless Communications*, 7(4):1400–1407, April 2008.
[49] H. Wang, Shuang Chen, and Yong Xie. An rfid-based digital warehouse management system in the tobacco industry: a case study. *International Journal of Production Research*, 48(9):2513 – 2548, 2010.
[50] R. Want. An introduction to rfid technology. *IEEE pervasive computing*, 5(1):25–33, 2006.
[51] R. M. Weber. Internet of things becomes next big thing. *Journal of Financial Service Professionals*, 70(6):43 – 46, 2016.
[52] H. Wu and Y. Zeng. Passive rfid tag anticollision algorithm for capture effect. *IEEE Sensors Journal*, 15(1):218–226, Jan 2015.
[53] F. Xia, Laurence T Yang, Lizhe Wang, and Alexey Vinel. Internet of things. *International Journal of Communication Systems*, 25(9):1101, 2012.
[54] Bo Yan, Chang Yan, Chenxu Ke, and Xingchao Tan. Information sharing in supply chain of agricultural products based on the internet of things. *Industrial Management & Data Systems*, 116(7):1397–1416, 2016.
[55] L. Yang, J. Han, Y. Qi, and Y. Liu. Identification-free batch authentication for rfid tags. In *The 18th IEEE International Conference on Network Protocols*, pages 154–163, Oct 2010.
[56] D. Zhang, J. Zhou, M. Guo, J. Cao, and T. Li. Tasa: Tag-free activity sensing using rfid tag arrays. *IEEE Transactions on Parallel and Distributed Systems*, 22(4):558–570, April 2011.
[57] Degan Zhang, Guang Li, Zhaohua Pan, and Yanpin Liang. A new anticollision algorithm for rfid tag. *International Journal of Communication Systems*, 27(11):3312–3322, 11 2014.
[58] R. Zhang, Y. Liu, Y. Zhang, and J. Sun. Fast identification of the missing tags in a large rfid system. In *2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pages 278–286, June 2011.

Edited by: Kashif Akhtar Saleem
Received: Jul 4, 2017
Accepted: Sep 9, 2017

DOI 10.12694/scpe.v18i4.1334
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 18, Number 4, pp. 361–373. http://www.scpe.org

MOORECUBE: A SCALABLE AND FLEXIBLE ARCHITECTURE FOR CLOUD COMPUTING DATA CENTERS ON MULTI-PORT SERVERS

AREZOO JAHANI AND LEYLI MOHAMMAD KHANLI *

Abstract. Networks provide the infrastructure of cloud computing data centers. Servers in data centers grow to respond to the growing user demands. Data center scalability and flexibility in the use of multi-port servers is a challenging issue. Heretofore, no method has been considered for scalability and flexibility issues. This paper proposes a new architecture called MooreCube that can increase network scalability and decrease network diameter as well as increase flexibility. MooreCube is a scalable and flexible architecture that each multi-port server directly connected to other servers via bi-directional links, without using any switch. Furthermore, MooreCube is a recursively defined architecture that uses Moore graph as Building Block (BB) structure and uses the hierarchical structure to meet high scalability. The paper proposes a multipath routing to increase fault tolerance and decrease links burden. MooreCube architecture compared with other switchless architectures that use reserved ports to increase scalability. The simulation results show MooreCube increase scalability and flexibility along with decrease the diameter of the network.

Key words: Data center network, Cloud computing, Network architecture, Multi-port servers, Scalability, Multipath routing.

AMS subject classifications. 68M14, 68M10

1. Introduction.

Cloud computing uses data center networking (DCN) as an infrastructure to provide services [1]. The powers of data centers should be enhanced with an increase in the service request and computational, storage and processing requirements [2, 3]. Therefore the architecture of data centers must be scalable and loosely as possible to add and/or subtract server/s to data centers [4, 5]. In addition, to exposure better service, the architecture of data centers should provide communication between servers that is possible by high-speed links, bilateral and optimized routing [6].

Adding new server/s to data center has some Issues. The first issue is the network flexibility that possible adding any new server to the data center. The former architectures have limitations in adding new servers. Because most of them use the reserved port and when the reserved ports completed, adding any new server is not possible [3, 7]. Cost is the second issue. Adding new server/s has cost like the price of adding the new link, switch, and router. Although it has some other issues like inefficient network development which limit the use of all server capacity because of inefficient development [8, 9]. Therefore considering these issues will help in designing new data center architecture. Design goals of the data center are; scalability, fault tolerance, latency, network capacity, simplicity, flexibility, and network configuration that will describe in detail [2, 32].

- Scalability: Ease of adding new server/s to data center networks. Scalability shows that adding any new server should not modify the DCN architecture (topology).
- Fault tolerance: The DCN should be accessible in presence of fault. Mostly, the lack of alternative routes and servers lead to irreparable damage.
- Latency: Delay in sending and receiving the message has a direct relation with network diameter (number of steps between source and destination). So latency in DCN architecture is important. The network diameter is most important factor in architecture and should not increase with enhancing network scalability.
- Network capacity: Total network capacities such as server's CPU, hard and link bandwidth should be accessible and using all network capacity should be possible.
- Simplicity: Ease of creating architecture is an important issue that must be considered. Use of former routing would be possible if the DCN architecture not modified in adding any new server/s.
- Flexibility: Flexibility can be defined in two ways. Network Flexibility and flexibility in using multi-port servers. Network Flexibility means the possibility of using small and very large architecture scale in presence of all network features. Flexibility in using multi-port servers as the name implies is possible using servers with a different number of ports.

*Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran. (a.jahani@tabrizu.ac.ir, l-khanli@tabrizu.ac.ir)

- Modularity: Recursive architecture has modularity concept. This kind of architectures, use a basic structure that repeated with adding any new server and known as modularity architectures.

Data center architectures include in three general categories: switch-centeric architecture, server-centeric architecture and switchless architecture [2, 6, 7]. First two categories have the switch in their architecture to connect servers. The only difference between them is the location of network routing. In a switch-centeric architecture, the switch is responsible for message routing in the network. But in server-centric architecture, servers set up network routing. The last one not uses the switch in DC architecture and connects servers directly using multi-port servers [2]. Using any switch in DC architecture has some advantages like; reducing the cost and increasing the network efficiency.

This paper proposes MooreCube as a new architecture that not use any switch to connect servers. This new architecture uses multi-port servers to connect servers directly with bi-directional links. MooreCube use Moore graph as a basic Building Block (BB) and hierarchical structure to increase network scalability. Moore graph is a regular graph that is known with degree and diameter [12, 13]. The paper uses multipath routing to increase fault tolerance and reduce the burden of links [14, 15].

The reminder of paper is organized as follow; section 2, studies previous solutions. Section 3 describes MooreCube architecture. Section 4, shows multipath routing and in continue, section 5 evaluates the MooreCube strategy and at last, section 6 express conclusion and future works.

2. Related works. Researchers are trying to find a way to properly connect servers to each other to reduce the network diameter and increase network scalability. Many researchers have been done activities in this area. In general, the DCN architectures divided into three categories include; switch-centric architecture, server-centric architecture, and switchless architecture. In switch-centric architectures, the switch is responsible for routing and this architecture mostly uses optical interconnections to send and receive messages [7, 11]. The first switch-centric architecture was proposed in 2008 called FatTree [16]. FatTree uses some switches and servers as Building block and increases the number of switches with adding any new server/s. FatTree has the three-level architecture that the servers are at the lowest level and switches are in two high levels of architecture. Depending on the number of servers that are at the lowest level, switches have the port. These layers are connected to each other in form of tree. However, each routing between servers in lowest level needs to use existing switches in the high level. In the worst case, all the pairs of servers are linked together. So, using the switches in this way, make switches as a bottleneck. Portland architecture [17], is another architecture that uses FatTree with a new send/receive protocol [18]. VL2 is another architecture that focuses on the use of server capacity and facilitates the routing with using of flat addresses. SWDC [23], Poincare [22], Scafida [21], S2 [20], Jellyfish [19] and RingCube [7] are also switch-centric architectures. All of the solutions use switch beside server in DCN architecture.

Server centric architecture also has a switch but use the servers as a router. DCell [24], BCube [25], Ficonn [26], FlatNet [27], HCN & BCN [28], Dpillar [29], SWCube [30], FleCube [31, 32], DCube [33], Fsquare [34] and sprintNet [35] are server-centric architecture. Using the switch in this kind of architecture may lead to bottleneck and increase cost and scalability of the network. The third category of DCN is switchless architecture whose does not use any switch to connect servers. But servers are directly connected to one another by the bidirectional link. Do not use the switch in DC architecture reduce the cost and increase the network efficiency [35]. Because in the absence of switch, the cost, and power that are needed to switch cares, are stored. Also if we do not have the switch, switches will not have downtime and network fault tolerance will be greater. Delay in the switch is also another reason for not using the switch. Because each switch has few microsecond delay to send and receive data. We can use the server instead of switch in DCN and increase network scalability [33].

CamCube [36], Smallworld [23], NovaCube [37] and FleCube [2] are switchless architecture. CamCube uses three-dimensional Torus structure for connectivity between the servers. This architecture needs 6-port servers in all circumstances. The Torus-3D indicated with k-array-3-Cube that the number of servers (S) computed with Eq. (2.1) and Diameter (D) computed with Eq. (2.2).

$$S = k^n = k^3 \tag{2.1}$$

$$D = \sqrt[3]{(k^n)} = \sqrt[3]{(k^3)} = k \tag{2.2}$$

In Eq. (2.1), S is the number of servers and k indicates the number of servers required in each row of CamCube architecture and n is the dimension of Torus architecture (in CamCube n=3). In Eq. (2.2), D is the network diameter. Smallworld is another architecture that presented to reduce the diameter of CamCube. This architecture in order to reduce network diameter, add a random number links to CamCube architecture. Results show that the network diameter is reduced by adding additional links [23], but the number of ports on each server must be greater than 6 and we cannot definitely determine needed ports on each server. Also, the randomly added links are only shortest path and may become a bottleneck. NoveCube is another architecture that adds new links but not randomly. This architecture found pair servers that are far from each others and then with adding new link connects them. So the diameter of the network will be decreased. But this approach also has the problems of previous work and needs more ports. SprintNet has the new way to connect servers to each other without use of switches [33]. This method is flexible and can be implemented with multi-port servers independent of the number of ports. The problem with this solution is its limited scalability due to server port number. For example, with 3-ports servers cannot connect more than 42 servers directly. Our presented MooreCube architecture solves port and diameter issue and is scalable and flexible with any number of server ports.

3. MooreCube architecture. In this section, physical structure and characteristics of MooreCube are expressed. In order to measure the diameter of the network, one-way routing described below to find the shortest path between servers.

Physical structure. MooreCube architecture uses multi-port servers in its structure. Multi-port servers connected to one another by bidirectional links. This architecture does not use any switch to connect servers.

Moore graph. MooreCube uses recursive structure and multi-port servers. The basic Building Block (BB) in this architecture is Moore graph. Moore graph is one of the most famous mathematical graphs, and these graphs may be used in any number of vertices [12, 13]. In fact, Moore graph is a k-regular ($k > 2$) graph that is displayed with two variable $n = (v, g)$. v represents the degree vertices in the graph and g is waist size of graph or a complete cycle (without duplicate vertices) of the graph. With these two variables, we can calculate the number of vertices in the graph (n) using Eq. (3.1).

$$n(v,g) = \begin{cases} (1+(v-1)^{((g/2)-1)} + v\sum_{r=0}^{(g-4)/2}(v-1)^r, & \text{if } g \text{ is even} \\ 1 + V\sum_{r=0}^{(g-3)/2}(v-1)^r, & \text{otherwise} \end{cases} \tag{3.1}$$

In Eq. (3.1), n is the number of vertices in the graph. v is vertices degree and g is waist of the graph. Fig. 3.1 outlines a number of Moore graphs with different vertices degree.

As mentioned in Fig. 3.1, the graph (n=4) is a Moore graph with the degree (v=3) and waist (g=3) that can be computed with Eq. (3.1). Fig. 3.1 indicates various Moore graphs with different vertices [38]. With compliance Moore graph on DCN, graph vertices will express the servers and edges will express link between servers. In this case, v (the degree of Graph) will indicate the number of needed ports (in BB) and g will indicate network girth. Calculate the network diameter with the use of g will be described in the following.

Moore graph can be different values for v and g. also the number of Moore graphs generated for each pair (v, g) can be more than one. For example, for the pair (3,5) we can generate nine different Moore graph. Moore graph shows the flexibility for any environment and can be used with any wire length.

Some of the specific values of v and g graphs created and has the certain name that was given to them by the mathematical researchers. Table 3.1 shows some of these names and specific Moore graphs.

Peterson graph is a special case of Moore graph with the vertex as v=3 and waist as g=5. The pair (3,5) can generate nine different Moore graphs as shown in Fig. 3.2 and has 10 servers in total that can be computed in Eq. (3.1) ($n(v, g) = 10$).

According to what is shown in Fig. 3.2, Peterson graph in all of 9 cases has the diameter equal to 2. This means that any two vertices that are far from each other have distance equal to 2. So we need a way to calculate the diameter of network using girth.

Theorem. Diameter of each regular graph is $D_{Basic} = \lfloor g/2 \rfloor$, if g indicates waist length of the graph. (D_{Basic} indicates the diameter of graph.)

Proof: Waist length in each graph is equal to the number of edges from each vertex to itself through the unique vertices. Since the graph is regular, this length for all graph vertices will be equal. For each vertex,

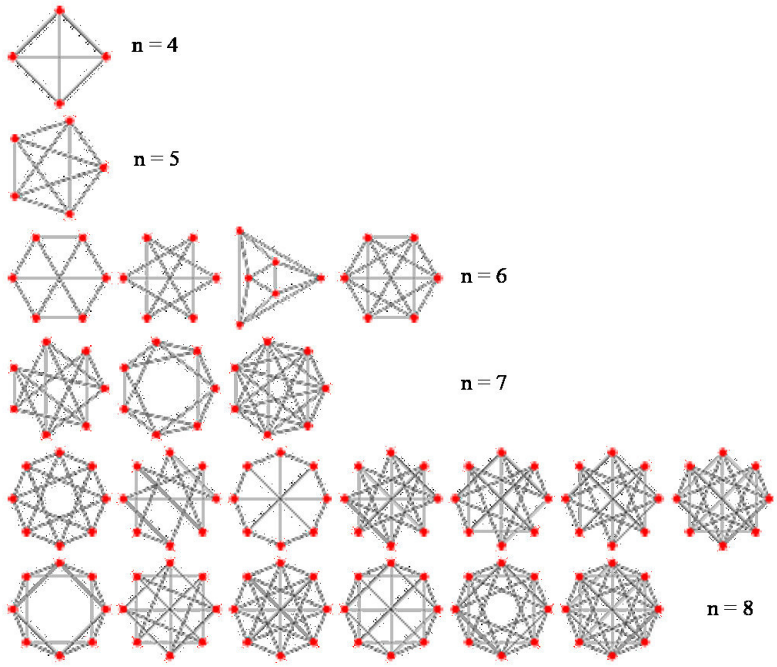


FIG. 3.1. *Moore graphs with different number of vertices*

TABLE 3.1
Specific Moore graph

(v,g)	**Name of Moore Graph**
(3,5)	graph Petersen
(3,6)	graph Heawood
(3,8)	graph Levi
(4,6)	graph Wong
(5,6)	graph order-4 generalized triangle
(7,5)	graph Hoffman-Singleton

a path that shows the number of edges from the vertex to itself has passed the farthest vertex of the graph and returns to the source vertex. So the waist graphs show a complete cycle through the farthest vertex of the graph. Obviously, the length of the waist can be halved to calculate the farthest vertex. As mentioned in Fig. 3.2(a), the distance from each vertex to farthest vertex in the graph is 2. So for different values of g, the network diameter is equal to low $g/2$ and prove that $D_{Basic} = \lfloor g/2 \rfloor$.

Opposite Theorem. For every regular graph with D_{Basic} diameter, waist length g calculated as $2D_{Basic}$ or $2D_{Basic} + 1$.

Peterson graph has vertex degree equal 3. This means if basic BB structure degree was 3, we need to three ports of multi-port servers. For network scalability, we can connect BB structures with links and in a hierarchical structure. Suppose the first case of Peterson graph is used as the basic structure of architecture. This basic structure has 10 servers that can be put k number of these structures over each other as showed in Fig. 3.3. For example, if we want to have 30 servers, we should 3 BB over each other.

As mentioned in Fig. 3.3, to scale the network, two extra port of each server is required to connect BB structures on top and bottom to each others. The proof shows that the number of ports per server in MooreCube is $v + 2$. v ports to connect to other servers in same BB and 2 port to connect to other BB on top or bottom of that.

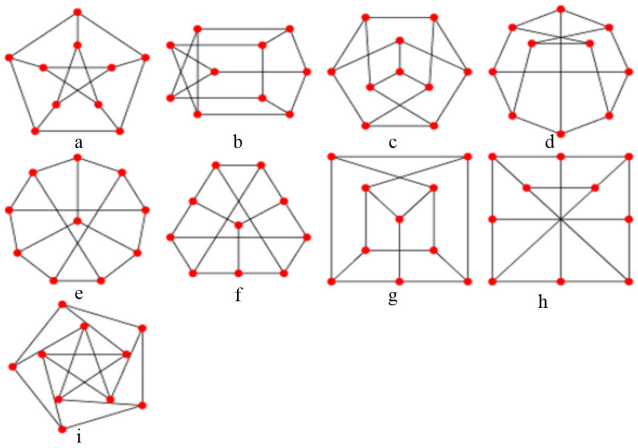


FIG. 3.2. *Moore graphs generated for the pair (3,5)*

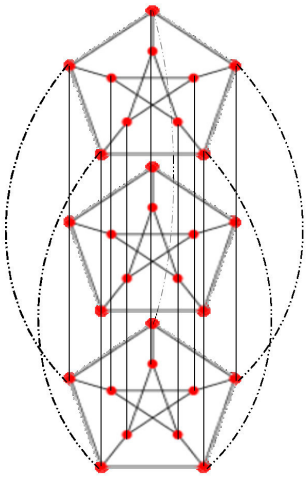


FIG. 3.3. *Peterson graph used as basic for a structure with 30 servers*

However, when mapping the network with Moore graph, at first we need to know how many ports are required for each server. It should be noted that the proposed architecture is capable to build with any number of ports which is an advantage and Indicator of Scalability. With having the number of ports that are required to each server, the basic structure can be found (v=number of ports-2). By choosing variable v, we can calculate needed servers for any value of g and selected the best and the most accepted structure. With having g and v, the BB structure will be determined by Eq. (3.1). Then with dividing the whole needed servers (S) to the number of servers in each BB can compute the number of BB that is required that is indicated in Eq. (3.2).

$$k = S/n(v, g) \tag{3.2}$$

In Eq. (3.2), k is the number of BB and S is the whole number of servers and $n(v, g)$ indicates the number of servers in BB. As shown in Fig. 3.3, the whole diameter of the network is different from BB diameter (D_{Basic}). Therefore it is necessary to calculate the overall diameter of the network. Overall diameter equal to the diameter of the BB network, plus the number of steps required to go from one structure to the farthest structure. Because every time there is a link that connects the two upper and lower structure, so pairs of far

TABLE 3.2
Calculate best waist length with degree=3

v	g	$n(v,g)$	$S/n(v,g)=k$	S	$D_{Basic}=\lfloor g/2 \rfloor$	$D_{Total}=D_{Basic}+\lfloor k/2 \rfloor$
3	1	1	30	30	1/2	16
3	2	2	15	30	1	8
3	3	4	7.5	30	1	4
3	4	6	5	30	2	4
3	5	10	3	30	2	3
3	6	14	2.14	30	3	4

structures has a length of $\lfloor k/2 \rfloor$. The overall diameter of the network is calculated by Eq. (3.3).

$$D_{Total} = D_{Basic} + \lfloor k/2 \rfloor \tag{3.3}$$

In Eq. (3.3), D_{total} is the overall diameter of the network, though D_{Basic} is BB diameter and k is the number of BB.

Practical example. Suppose we have 30 servers and each server has 5 ports. In order to find the number of required BB, first, we should compute the degree of the basic graph. Because the servers have five ports, two ports of them will be reserved to connect BBs together. Then we have three ports for BB graph and v=3. For v=3, we can consider different waist lengths. Table 3.2 shows the number of servers used in any structure with different g. in this table, D_{Total} can be calculated for each g. so the most appropriate structure will be selected. It is obvious the row with lowest D_{Total} in the table is the best structure. But no need to calculate all values for all of g. because g and D_{Total} has a regular trend. D_{Total} will be reduced with increasing g and then enhanced when passed the best value of g. this regular trend helps us to choose the most suitable amount of g. in Table 3.2, the lowest of D_{Total} belong to the fifth row. So the value of $g = 5$ will be chosen that has 10 servers in the structure of the BB.

According to Table 3.2, D_{Total} with different g, will be declined at first and then with passing the best value of g, start to enhanced. So choose the most optimal value for the variable g easily computed with the comparison of values in the D_{Total} column.

4. Routing. MooreCube structure constructed of many BB and each BB constructed of many servers. Routing in each BB can be done by broadcast shortest path (BSP) algorithm. It is essential to note that routing algorithms run once before DCN running. So DCN uses the results of routing algorithm. In fact, all routes should be saved in routing tables for future use. So the complexity of routing algorithm will not have any effect on the routing between data centers and most of the routing algorithm can be easily used for routing. Finding the shortest path with BSP is very simple algorithm where each server in order to find the shortest route to the destination server, first sends a message includes destination server name to all neighbors. The neighbors who received this message sends the message to their neighbors. This process continues and if the message received by destination server, process terminated and messages come back to the source server and collect the intermediate servers as a route from source to destination. The source has received several messages from different directions to pass each of which have the shortest path as the best path is selected. Routing in each pair of BB is carried out by a link between them. Each server who wants send a message to a server in another BB, there are two choices:

- Use the link that connects the source to destinations neighbor (in destination BB).
- Use the link that connects destination to sources neighbor (in source BB).

Routing procedure between each source and the destination indicated in the algorithm 1.

As indicated in the algorithm 1, this algorithm first check if source and destination are the neighbor. If they were the neighbor, the link between them returned as route (rows 1-3). Otherwise, select one of the links between two BB as the intermediate link in order to find the shortest path (row 4). Then depending on which link is chosen, calculate the shortest path using BSP algorithm (rows 5-9).

Algorithm 1 BSP(src,dst)

Require: *src*: source server;
dst: destination server;
src and *dst* located in BB;
1: **if** *src* and *dst* are adjacent **then**
2: return (*src* , *dst*);
3: **end if**
4: obtain (link(*src* , s1) between BB *src* and *dst* — link(s2 , *dst*) between BB *src* and *dst*);
5: **if** link(*src* , s1) **then**
6: return (BSP(*src* , s1) , *dst*);
7: **else**
8: return (*src* , BSP(s2 , *dst*));
9: **end if**

MooreCube features.

- **Scalability:** MooreCube scalability enhances by increasing the number of ports (server ports) as well as network diameter. According to Eq. (3.2), the number of required BB is equal to $k = S/n(v, g)$. So the number of servers depends on k and the number of ports in BBs servers. As a result, the total number of servers in each layer of the hierarchical structure will be calculated by Eq. (4.1).

$$S_k = k \times n(v, g) \tag{4.1}$$

  Eq. (4.1) indicates the number of possible servers in each layer of hierarchical structure k. S_k is the number of servers for all k layer. K is the number of required BB and $n(v, g)$ indicates number of servers in each BB. Increase the number of servers with increasing variable k in Section 5 and will be shown in experiments related to scalability.
- **Flexibility:** Flexibility is one of the most important features in MooreCube architecture. The purpose of the flexibility is using architectural with the different number of ports. MooreCube is capable of using multi-port servers in architecture. Although the architecture FleCube [2] also was dealt with this issue in 2015. But FleCube focused on only flexibility and scalability was not adhered to. For example, when using FleCube architecture and 3-port servers we cannot have more than 42 server connection and adding more servers is not possible in this architecture. Therefore, the proposed solution unlike previous methods focusing on both flexibility and scalability.
- **Network diameter:** Network diameter indicates the number of steps from a source server to farthest server on the network. According to section (3.1.1) and proved theorem we know that the network diameter can be determined by variable g and the overall diameter of the network is $D_{Total} = D_{Basic} + \lfloor m/2 \rfloor$.
- **Bidirectional bandwidth:** Network bandwidth indicates data transfer rate on network connections. The criterion is the most important criteria for determining the speed of a network. Accordingly, bi-directional bandwidth indicates data transmission rate by network connections. MooreCube architecture has the bidirectional bandwidth. Eq. (4.2) shows the bidirectional bandwidth of MooreCube architecture.

$$v + 2 \leq bidirectional - bandwidth \leq n(v, g) \tag{4.2}$$

  In Eq. (4.2), v is the number of ports in each BB.
- **Theorem:** Bidirectional bandwidth in MooreCube structure is accordance with Eq. (4.2).
  Proof: To calculate the bidirectional bandwidth, the network should be divided into two equal part. The simplest way to divide MooreCube is to divide it into two separate BB that have equal servers. In this case, the link between these two parts will be equal to the number of servers in each BB unit. all servers of two connected BB have a link together. So at best case, bidirectional bandwidth is equal to

the number of servers in BB. In the worst case, there is only a BB should be divided into two equal parts. In dividing time, if the number of servers is even, we can divide them simply in two parts and it is clear that only $v + 2$ link between these two parts can connect parts to each others. Otherwise, if the number of servers is odd, we can omit one of them and use the earlier case to calculate network diameter.

- **Links and cost:** Given that more links in any architecture, increase fault tolerance on the network. But connecting any two servers is not possible. Because it led to wiring problems and eliminate the flexibility and scalability of the network. So we should make a trade-off between the number of links and scalability, flexibility and wiring problems. The number of links should be low as much as possible and next to it the flexibility and scalability of the network should be high as much as possible. MooreCube has the simple structure and adding any new server do not need much cost, because not need any switch and do not change BB structure. In addition, the number of links in each BB is less than the number of links in a complete graph and this is easily measurable by the number of used server ports.
- **Number of parallel disjoint paths:** According to section 4 of this article, the following is stated, it can be concluded that the number of parallel disjoint paths is equal to $v + 2$. The proof is given below.
- **Theorem:** The number of the parallel disjoint path in MooreCube is $Pk = v + 2$ (if Pk be the number of the path).
  Proof: Each server like a and b has set of neighbors such as $seta$ and $setb$. We know $|set_a| = |set_b| = v + 2$. These two elements can have three types of communication: (1) a and b are neighbor and $a \in set_b and b \in set_a$. So a and b has a link that connects them. (2) a and b have a common neighbor. So the number of common neighbors is $|set_a \bigcap set_b|$. In this case, a and b has a path with the transition from middle neighbors. (3) Each server in $seta$ belong to a BB and $setb$ belong to BB too. These two BB can be equal or not, in every case a and b have a link/path and can connect to each other. So prove that a path is between a and b with the transition from middle neighbors and $P_k = v + 2$.

5. Multipath routing. In order to have several paths which are parallel and disjoin, disjoint parallel path (DPP) is provided. DPP can select multiple paths between each pair of source and destination which has less congested links and uses all link capacity. So using DPP, increase bi-directional bandwidth efficiency and protect network failure.

DPP algorithm uses BSP to find disjoint parallel paths. But using BSP, may led to loop or common link paths in the network and we know this kind of paths which have common link or loop can cause network congestion and failure. So all of the parallel paths should be disjoint with any common links. In order to have such routes, we should change slightly BSP algorithm. This modified algorithm is shown CBSP. CBSP has three entrance; source, destination and limitation set that includes some server and CBSP should find the shortest path between source and destination without violating from limitation servers. The procedure is like that section (2.1.3) but source server should send limitation set when sending a broadcast message to their neighbors. Each intermediate server with receiving a message, check the content of the collection, if its id is found within the limitation set, it terminated and not send the broadcasting message. Algorithm (2) indicates DPP with use of CBSP (improved BSP).

Algorithm 2 calculate the disjoint parallel path for each pair of source and destination. This algorithm creates set u with neighbor servers id of the source server. Then if the source and destination server are the neighbor, return their link (rows 1-3). Otherwise, add source id to limitation set and send broadcasting the message (row 4). Each server can have parallel path depend on a number of ports, therefore for every server calculate paths based on their neighbors (row 5-8).

Then all intermediate servers added to limitation set (row 9) in order to next parallel path do not use this server as middle servers.

6. Evaluation results. This section evaluates MooreCube with three kinds of tests includes network diameter, flexibility, and scalability. In all tests, the proposed MooreCube and other compared solutions implemented in MATLAB software, version 7.14.0 (R2-15a) on a computer with Intel Core i5 Duo 2.53 GHz, 4 GB Memory and Windows 7 x86 enterprise.

MooreCube compared with CamCube [34] and FleCube [2] solutions. CamCube uses three-dimensional Torus structure and focuses on scalability and FleCube focus on flexibility.

Algorithm 2 DPP (src, dst)

Require: *src*: source server;// *dst*: destination server;// *limitation*: set of servers that should not be in routing// *u*: set of adjacent server of *src*

if *src* and *dst* are adjacent **then**
  return (*src* , *dst*);
end if
add *src* to limitation
while (*u* has unselected server) **do**
  *p*1=randomly get one unselected server of *u*
  set *p*1 status to select
  return (route=(*src*,CBSP(*p*1,*src*,limitation))
  add all internal servers of route to limitation
end while

In following this section, an experiment done for measure delay in sending and receiving message using DPP and results are given below.

6.1. Network diameter. In this experiment, the diameter is calculated with increasing number of servers. Server numbers are from 10 to 105. In each stage, based on a number of servers, appropriate network diameter is selected. The results are shown in Fig. 6.1.

As shown in Fig. 6.1, when 6-port servers increase, network diameter in CamCube increase exponentially. FleCube has better network diameter but focuses only on flexibility. Our proposed MooreCube has lowest network diameter. Because MooreCube uses Moore graph and can use any server ports and can have scalability using the hierarchical structure.

6.2. Flexibility. Flexibility means useing of architecture with any number of ports in the server and the architecture must not depend on port and should be usable regardless of server ports. MooreCube is constructive with any number of server ports. This experiment shows MooreCube flexibility in compress of FleCube and CamCube. This experiment measures the maximum number of servers that each strategy can connect with a specified port number. The experiment results are shown in Fig. 6.2.

As indicated in the Fig. 6.2, CamCube use only 6-port servers and can connect any number of servers with

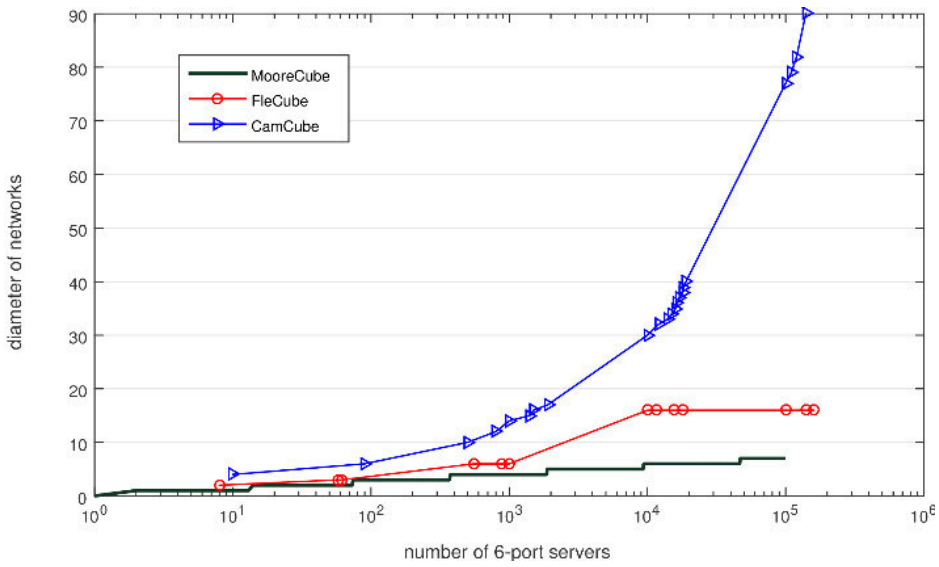


FIG. 6.1. *Network diameter (6-ports server)*

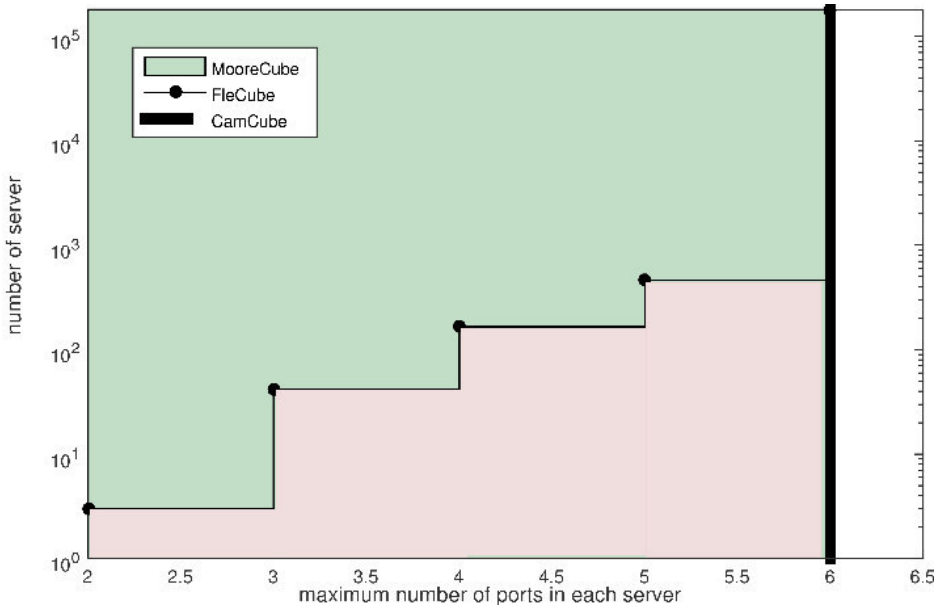


FIG. 6.2. *The maximum number of servers used in the number of ports*

this port number. So CamCube does not have any flexibility in using server ports. FleCube has more flexibility than CamCube and can use any number of ports. The only problem with this approach is that by choosing the number of required ports, we cannot add new servers more than the certain number that is clear at first and we can calculate at first. For example, with 3-port servers in FleCube, the maximum number of servers that can connect is 42. So it has less flexibility in using server ports. But MooreCube is more flexible than others and can use any number of server ports.

6.3. Scalability. Scalability means the maximum number of servers used with any given network diameter. In the experiment, network diameter increased from 3 to 33 and in each stage, a maximum number of servers that can connect is calculated. Results are shown in Fig. 6.3.

As indicated in Fig. 6.3, CamCube do not have any scalability with increasing network diameter. FleCube is more scalable than CamCube. But MooreCube is more scalable than CamCube and FleCube. Because uses Moore graph and can construct with any number of server ports.

6.4. Delay in routing. This experiment calculates link delay in sending and receiving the message. Delay is calculated in two modes; with congestion and without congestion. The results are shown in Fig. 6.4.

As indicated in Fig. 6.4, if links have no congestion, the delay is constant and delivery time of packets not change. It means routing algorithm correctly runs and is able to use different routes which are in the routing table. If links have congestion, the delay will be enhanced with increasing flow. The increase in delay is due to static routing in data centers. Static routing, run before using DCN and all of the paths stored in routing tables.

7. Conclusion and future work. In this paper, we proposed MooreCube as a DCN architecture that is scalable and flexible. MooreCube does not use any switch and connects servers directly. So it stores the cost of the purchase and maintenance of the switch and has better performance than switch-centric architectures. MooreCube has more scalability and flexibility and less network diameter. MooreCube uses DPP for routing in DCN. DPP calculate parallel paths depends on server ports. Parallel paths distribute flow between links and prevent network failure.

As future work we intend to use of compound graph as BB to decrease network diameter. This paper focus on scalability and flexibility, as a future work, we want to propose a dynamic parallel path routing that runs simultaneously with the network using. Decreasing delay can be another future work. Producing parallel paths

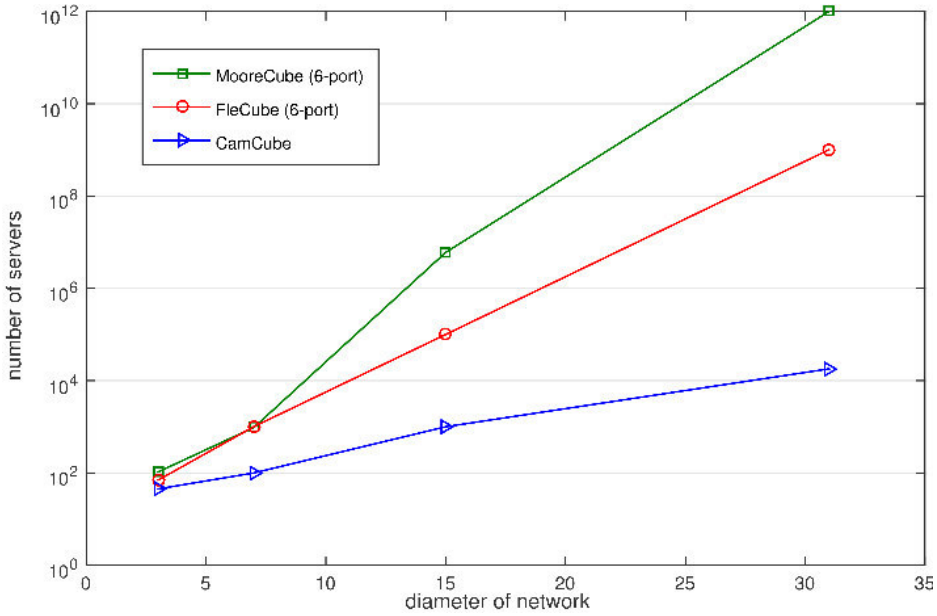


FIG. 6.3. *Scalability on network diameter*

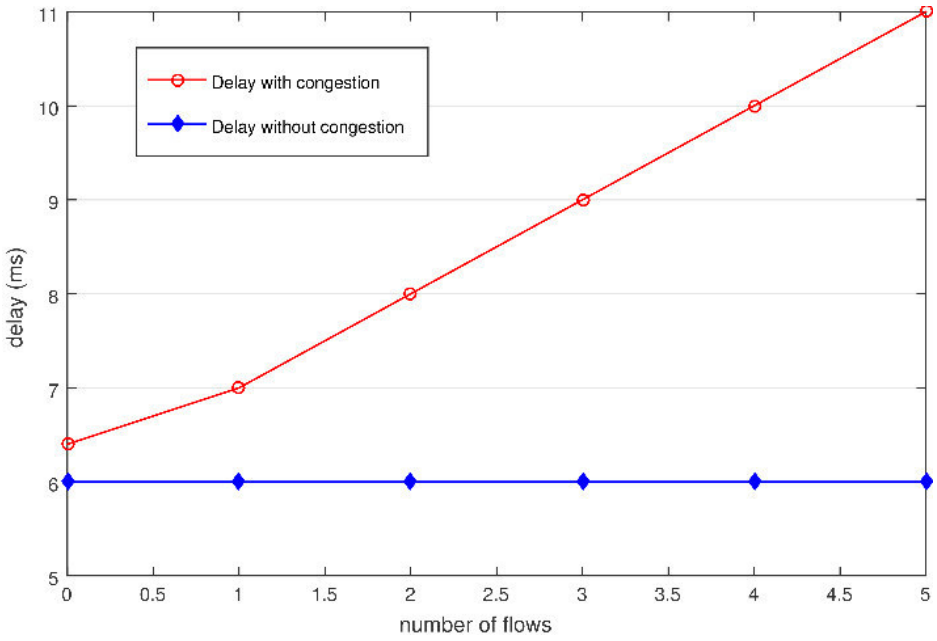


FIG. 6.4. *Average delay and number of passed flow in MooreCube*

with the same number of links is another future work.

REFERENCES

[1] W. TIAN AND Y. ZHAO, *Resource modeling and definitions for cloud data centers*, Chapter 3, Optimized Cloud Resource Management and Scheduling, 2015, pp. 51-77.

[2] D. LI, Y. SHEN AND K. LI, *FleCube: A flexibly-connected architecture of data center networks on multi-port servers*, Computer

Communications, 70 (2015), pp. 1-10.
[3] B. Wang, z. Qi, R. Ma, H. Guan and V. Vasilakos, *A survey on data center networking for cloud computing*, Computer Networks, 91 (2015), pp. 528-547.
[4] G. Wu, H. Gu, K. Wang, X. Yu and Y. Guo, *A scalable AWG-based data center network for cloud computing*, Optical Switching and Networking, 16 (2015), pp. 46-51.
[5] Y. Shen, K. Li, and W. Shi, *Advanced topics on cloud computing*, Journal of Computer and System Sciences, 79 (2013), pp. 1199, 2013.
[6] T. Wang, Z. Su, Y. Xia, J. Muppala and M. Hamdi, *Designing efficient high performance server-centric data center network architecture*, Computer Networks, 79 (2015), pp. 283-296.
[7] X. Yu, H. Gu, Y. Yang and K. Wang, *RingCube An incrementally scale-out optical interconnect for cloud computing data center*, Future Generation Computer Systems, 54 (2016), pp. 41-51.
[8] S. Fu, B. Wu, X. Jiang, A. Pattavina, H. Wen and H. Yu, *Switch cost and packet delay tradeoff in data center networks with switch reconfiguration overhead*, Computer Networks, 87 (2015), pp. 33-43.
[9] M. Uddin, Y. darabidarabkhani, A. Shah and J. Memon, *Evaluating power efficient algorithms for efficiency and carbon emissions in cloud data centers: A review*, Renewable and Sustainable Energy Reviews, 51 (2015), pp. 1553-1563.
[10] S. Filiposka and C. Juiz, Community-based complex cloud data center, *Physica A: Statistical Mechanics and its Applications*, 419 (2015), pp. 356-372.
[11] A. Erickson, A. Stewart, J. Navaridas and A. Kiasari, *Star-replaced networks: A generalised class of dual-port server-centric data center networks*, Computer Networks, 2015, pp. 1-30.
[12] M. Miller and J. Siran, *Moore graphs and beyond A survey of the degreediameter problem*, University of Newcastle, The Electronic Journal of Combinatorics, 20 (2013), pp. 1-92.
[13] E. Ducey, *On the critical group of the missing Moore graph*, Discrete Mathematics, 340 (2017), pp. 1104-1109.
[14] T. Baker, B. Dawsari, H. Tawfik, D. Reid and Y. Ngoko, *GreeDi: An energy efficient routing algorithm for big data on cloud*, Ad Hoc Networks, 35 (2015), pp. 83-96.
[15] Z. Zhang, W. Hu, T. Ye, W. Sun, L. Zhao and K. Zhang, *Routing and spectrum allocation in multi-ring based data center networks*, Optics Communications, 360 (2016), pp. 25-34.
[16] M.A. Fares, A. Loukissas and A. Vahdat, *A Scalable, commodity data center network architecture*, Acm Sigcomm, 2008, pp. 1-12.
[17] R. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya and A. Vahdat, *PortLand: A Scalable Fault-Tolerant*, Sigcomm Proceedings of the ACM Sigcomm, Conference on Data Communication, Newyork, USA, 2009, pp. 39-50.
[18] A. Greenberg, J. Hamilton and N. Jain, *VL2: a scalable and flexible data center network*, Sigcomm Proceedings of The ACM Sigcomm, Conference on Data Communication, New York, USA, 2009, pp. 51-62.
[19] A. Singla, Ch. Hong, L. Popa and B. Godfrey, *Jellyfish: networking data centers randomly*, NSDI'12 Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, Berkeley, USA, 2012, pp. 1-6.
[20] Y. Yu and C. Qian, *Space Shuffle: a scalable, flexible and high bandwidth data center network*, ICNP '14 Proceeding of the 2014 IEEE 22nd International Conference on Network Protocols, 2014, pp. 13-24.
[21] L. Gyarmati and T. Trinh, *Scafida: a scale-free network inspired data center*, ACM Sigcomm Computer Communication Review, 40 (2010), pp. 4-12.
[22] M. Csernai, A. Gulyas, A. Korosi, B. Sonkoly and G. Biczok, *Incrementally upgradable data center architecture using hyperbolic tessellations*, Computer Networks, 57 (2013), pp. 1373-1393.
[23] J. Shin, B. Wong and E. Sirer, *Small-World Datacenters*, SOCC '11 Proceedings of the 2nd ACM Symposium on Cloud Computing, Newyork, USA, 2011, pp. 1-13.
[24] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang and S. Lu, *DCell: A Scalable and Fault-Tolerant Network*, ACM Sigcomm Computer Communication Review, 38 (2008), pp. 75-86.
[25] C. Guo, G. Lu, D. Li, L. Wu, X. Zhang and Y. Shi, *BCube: A high performance, Server-centric Network*, ACM Sigcomm Computer Communication Review, 39 (2009), pp. 63-74.
[26] D. Li, H. Wu, K. Tan, Y. Zhang and S. Lu, *FiConn: using backup port for server interconnection in data centers*, IEEE Infocom, 2009, pp. 2276- 2285.
[27] D. Lin, Y. Liu, M. Hamdi and J. Muppala, *FlatNet: towards a flatter data center network*, Globecom Next Generation Networking and Internet Symposium, 2012, pp. 2499-2504.
[28] D. Guo, T. Chen, D. Li, M. Li, Y. Lio and G. Chen, *Expandable and cost-effective network structures for data centers using dual-port servers*, IEEE Transaction on Computers, 62 (2013), pp. 1303-1317.
[29] Y. Liao, J. Yin, D. Yin and L. Gao, *DPillar: dual-port server interconnection network for large scale data centers*, Computer Networks, 56 (2012), pp. 2132-2147.
[30] D. Li and J. Wu, *On the design and analysis of Data Center Network architectures for interconnecting dual-port servers*, Infocom IEEE Conference on Computer Communication, 2014, pp. 1851-1859.
[31] Li, Deshun, Yanming Shen, and Keqiu Li, *FleCube: A flexibly-connected architecture of data center networks on multi-port servers*, Computer Communications 77 (2016): 62-71.
[32] A. Gawanmeh,, and A. Alomari, *Challenges in formal methods for testing and verification of cloud computing systems*, Scalable Computing: Practice and Experience, 16 (2015): 321-332.
[33] D. Guo, C. Li, J. Wu and X. Zhou, *DCube: A family of network structures for containerized data centers using dual-port servers*, Computer Communications, 53 (2014), pp. 13-25.
[34] D. Li, J. Wu, Z. Liu and F. Zhang, *Dual–centric data center network architectures*, in Proceedings of IEEE ICPP_15, 2015.
[35] T. Wang, Z. Su, Y. Xia, J. Muppala and M. Hamdi, *SprintNet: A high performance server-centric network architecture*

for data centers, Computer Networks, 79 (2015), pp. 283-296.
[36] H. Libdeh, P. Costa, A. Rowstorn, G. Shea and A. Donnelly, *cam cube: symbiotic routing in future data centers*, ACM Sigcomm Computer Communication Review, 40 (2010), pp. 51-62.
[37] T. Wang, Z. Su, Y. Xia, B. Qin and M. Hamdi, *NovaCube: A low latency torus-based network architecture for data centers*, IEEE Global Communications Conference (Globecom), Austin, TX, 2014, pp. 2252-2257.
[38] A. Hoffman and R. Singleton, *On Moore graphs of diameter 2 and 3*, IBM Journal of Research and Development, 4 (1960), pp. 497-504.

Edited by: Amjad Gawanmeh
Received: Jul 1, 2017
Accepted: Oct 27, 2017

DOI 10.12694/scpe.v18i4.1335
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 18, Number 4, pp. 375–385. http://www.scpe.org

DATA FLOW ANALYSIS OF MPI PROGRAM USING DYNAMIC ANALYSIS TECHNIQUE WITH PARTIAL EXECUTION

K. B. MANWADE* AND D. B. KULKARNI†

Abstract. Message Passing Interface (MPI) is a dominant parallel programming paradigm. MPI processes communicate with each other by sending or receiving messages through communication functions. The application's communication latency will be less if processes are scheduled on nearest cores or nodes and communication latency will be more if processes are scheduled on farthest cores or nodes.The communication latency can be reduced by using topology-aware process placement technique. In this technique, MPI processes are placed on the nearest cores if they have more communication between them. To find the communication pattern between processes, analysis of MPI program is required. Various techniques like static, symbolic and dynamic analysis are available for finding communication pattern of MPI program. These techniques are either taking more time for analysis or fail to find correct communication pattern. In this paper, we have proposed DAPE (Dynamic Analysis with Partial Execution) technique for analysis of MPI program, which finds correct communication pattern in less time as compared to existing techniques. The experimental results show that the proposed technique outperforms over the existing techniques.

Key words: Topology-aware process placement, Communication pattern of MPI program, Dataflow analysis of MPI program

AMS subject classifications. 15A15, 15A09, 15A23

1. Introduction.

MPI [1] is a dominant message passing paradigm used for parallel programming. Formerly it was used for computer systems where every node was having a single processor on which a single process could be executed. The development of multi and many core technologies made scheduling of MPI processes more difficult. Nowadays, scheduling of more than one process on multi and many core nodes is possible. For such nodes, data localities need to be considered. If the two processes have more communication between them then scheduling them on two nearest cores improves the performance of a program. This leads to the topology-aware process placement [2] concept. The topology aware process placement involves three steps 1) finding communication pattern or topology of a program 2) finding architectural details or topology of nodes 3) scheduling processes on nodes. To find a communication pattern of the program, its analysis is required. In literature, various techniques such as static analysis, symbolic analysis, and dynamic analysis [3] [4] [5] are proposed for the analysis of MPI program. The data structure DFG (Data Flow Graph) represents the flow of data in the program through different functions and processes. Construction of DFG for MPI program is challenging because of its dynamic and SPMD nature. Processes of MPI program can communicate through communication functions in which receiver, sender, tag, communicator, size of the message, an address of message buffer, MPI data type, root process for collective function etc are mentioned. If these functions contain variable or MPI "wildcard" variables (MPI_ANY_SOURCE, MPI_ANY_TAG etc) then identifying the correct communication pattern using static analysis is difficult. The static analysis technique takes less time for analysis but communication patterns may not be correct. Therefore to find correct communication pattern dynamic analysis of the program should be performed. In this technique first run of the program is required which takes more time for the large program. The symbolic analysis is used at compile time and symbolic presentation of each instruction is required. In this paper, we proposed a new technique called dynamic analysis with partial execution, which detects correct communication pattern in less time as compared to existing techniques.

This paper is organized as follows: Section 2 presents work related to MPI program analysis. Section 3 reviews the concept of MPI program analysis, techniques of program analysis and motivation of the proposed work. The concept of partial execution and its assumptions are presented in section 4. The details of proposed technique are given in section 5. The performance comparison of the proposed technique with existing techniques is presented in section 6. Finally, in section 7, the conclusions are noted.

*Ph.D.Research Center:-Department of Computer Science and Engineering, Walchand College of Engineering, Sangli, Maharashtra, India. University:- Shivaji University, Kolhapur, Maharashtra, India. (mkarveer@gmail.com).

†Department of Information Technology, Walchand College of Engineering, Sangli, Maharashtra, India. (d_b_kulkarni@yahoo.com).

2. Related work. In [6], authors have applied the concept of static analysis to MPI program. They have constructed CFG (Control Flow Graph) for MPI programs, then the graph is converted into MPI-CFG where MPI communication function calls are analyzed and corresponding lines are joined between the nodes of CFG. In an analysis of MPI communication function sender, receiver, tag, communicator parameters etc are considered. They have matched sender and receiver by using three categories; explicitly identifier of the process, a variable containing process identifier and MPI "wildcard" variables for process identifier. To match sender and receiver processes, the first case uses constant identifier, the second case uses substitution values and the third case uses annotation on MPI-CFG graph. In [7], authors have constructed CFG for MPI program and by using fan IN, fan OUT notation the communication pattern of processes have been identified. For example, if sender process X and receiver Y are mapping in the record of both IN and OUT then it is assumed that both X and Y have communication between them otherwise not. For matching the sender and receiver process approximation method is used in [8]. Two approaches are used for this purpose; constant propagation and static slicing of CFG for non-constant propagation in MPI functions.

In [9] authors used two phases of analysis like (i) static pass to check whether the number of completed calls is matching with non-blocking calls (ii) static analysis to check whether the sequence of all collective calls is matched with all possible execution paths. These two phases detect an incorrect collective pattern in MPI program which leads to deadlock. The instrumentation overhead of these phases is very low but its functionality is limited to detection of deadlock in the program.

In [10] authors have used parallel version of static analysis with a fixed number of processes and they have extended functionality of Fuse framework to parallel implementation. The compositional approach is used which extend sequential data flow analysis to work with MPI program.

Dynamic analysis tools like ISP [11], DAMPI [12] are available for analysis of MPI applications. In these tools, program communication pattern is constructed by matching sender and receiver of the messages. These tools explore the non-determinism in the program but are used for performance analysis only. These tools are not used for detecting communication pattern of the processes. Also, tools like MPIPP [13] use dynamic analysis. The communication pattern determined by profiling MPI program takes more time for analysis. The MPI program profiling time is reduced in FACT [14] tool by using time-slicing method but it is facing the problem of non-determinism.

In [15] authors, reduced original program to program slice through static analysis and execute the program slice to acquire communication trace. The program slices preserve all variables and statements in the original program. For sliced program compile time analysis is performed. They have implemented FACT tool and evaluated with seven NBP programs as well as the sweep3D program.

In [16] authors implemented MOPPER deadlock detection tool which takes MPI program as input and produced deadlock detection as output. MOPPER first compile MPI program and then execute it using ISP tool. The ISP tool outputs a canonical trace of the input program along with the matches-before partial order. MOPPER then computes the over-approximation as match-setapp. Then both match-before and match-setapp are passed to SAT solver to find the deadlock in the communication.

The symbolic analysis method is used for MPI program analysis in [17] [18]. All MPI instructions are represented as a sequence of symbols by using 'instruction type'. By using this concept message blocks are identified. Then using symbolic representation CFG is constructed for the program. The combination of both control and data flow graph (CDFG) is used in [19] for MPI program along with SUIF [20] compiler framework. They have identified three types of communication patterns such as static, persistent and dynamic. To identify communication pattern of application symbolic analysis technique is used in TASS [21].

In this paper, we have proposed a novel method based on partial execution of a program which overcomes the limitations of existing techniques.

3. Techniques For MPI program analysis. The taxonomy of MPI program analysis techniques is shown in figure 3.1.

3.1. Static Analysis of Programs. Static analysis technique predicts program behavior like correctness of the program, control flow in the program, logical errors in the program etc without executing the program. In this technique, control flow graph is generated with help of control flow instructions in the program. By matching send instruction with corresponding receive instruction in the control flow graph, the communication

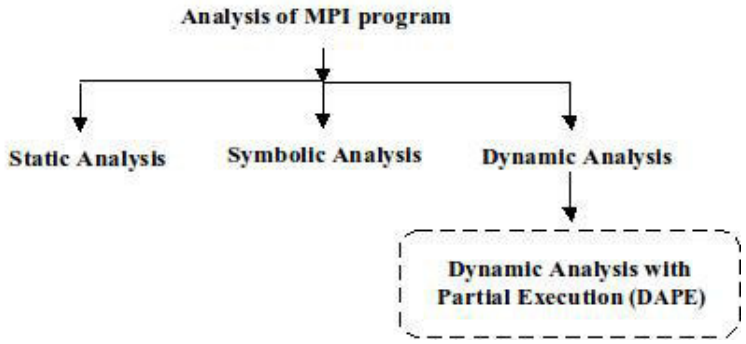


FIG. 3.1. *Techniques for MPI program analysis*

pattern of the program is found. The time efficiency of this technique is more as it finds a communication pattern without executing MPI program. This technique finds communication pattern in less time. It fails to find correct communication pattern in case of following conditions:

1. MPI_Send(...,...,...,dest,...,...)/ MPI_Recv(...,...,...,src,...,...):
   where 'dest' and 'src' are the variables and their values will be known at runtime.
2. MPI_Recv(...,...,...,MPI_ANY,...,...)/ MPI_Recv(...,...,..,..,MPI_ANY_TAG,..):
   where source of message and tag are indicated by "wildcard" variable 'MPI_ANY' and 'MPI_ANY_TAG' respectively whose value will be known at runtime.
3. Related messages:
   If the process sends a message to any process immediately after receiving a message from another process, then the two messages are related with happened before relation and its communication pattern will be decided at runtime.

3.2. Dynamic Analysis of Programs. In dynamic analysis technique, to find communication pattern of a program, instrumentation instructions are added to the program. The modified program is executed and execution traces are recorded. After analyzing execution traces, a communication pattern of a program is found. As communication patterns are found by executing a program, the discovered communication pattern is more accurate. The time efficiency of this technique is less as compared to static analysis technique.

3.3. Symbolic Analysis of Programs. In this technique instead of actual input, the symbolic constant inputs are given to the program. By using symbolic input all possible conditions in the control flow instructions are checked to verify the execution flow of the program. Because of non-determinism in the verification process, more time is required as compared to static analysis. It fails to discover correct communication pattern if variables are used in communication function.

3.4. Summary of existing techniques. To schedule processes using topology-aware process placement, a technique for finding correct communication pattern of a program in less time is required. Existing techniques are either taking more time for program analysis or fail to find correct communication pattern. To detect correct communication pattern in less time, we have proposed a DAPE technique for analysis of MPI program. The details of partial execution and its algorithm are given in section 4 and 5 respectively. Advantages and disadvantages of the existing technique are summarized in Table 3.1.

4. Analysis of MPI program.

4.1. Partial execution of MPI program. To understand the concept of partial execution, consider following MPI program (Listing 1) which contains four types of instructions.

a) Control flow [Line Number 9 and 15]
b) Computation [Line Number 8, 11, 13, 17, 19,21]
c) Communication [Line Number 12 and 18]
d) Declaration.

TABLE 3.1
Comparison of MPI program analysis techniques

Sr. No.	Technique	Advantages	Disadvantages
1	Static analysis	1.It is scalable as actual program execution is not required. 2.It takes less time as compared to other techniques.	1.Automation of this technique is difficult as program annotation by the user is required. 2.It cannot handle "wildcard" variables in MPI_Recv() function.
2	Dynamic analysis	1.A communication race conditions which are arising from "wildcard" variable in MPI_Recv() function are handled. 2.As it executes the entire program, it explores all possible behavior of a program.	1.It takes more time for analysis as the complete execution of a program is required. 2.It identifies communication patterns specific to a particular input.
3	Symbolic analysis	1.It verify properties of all possible behavior of MPI program 2.As input is symbolic value, it explores program behavior independent of input value	1.For its implementation sophisticated theorem proving technique and symbolic interpretation is required. 2.It cannot scale beyond a relatively a small number of processes.

LISTING 1
The sample MPI program

```
1. int main(int argc, char * argv[])
2. {
3.  int rank, numprocs;
4.  char buffer[20];
5.  MPI_Init(&argc, &argv);
6.  MPI_Comm_size(MPI_COMM_WORLD, &numprocs);
7.  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
8.  ..................computation
9.  if(rank%2==0)
10. {
11.    ..................computation
12.    MPI_Send(&buffer,10,MPI_CHAR,rank+1,tag,MPI_COMM_WORLD);
13.    ..................computation
14. }
15. else
16. {
17.    ..................computation
18.    MPI_Recv(&buffer,10,MPI_CHAR,rank-1,tag,MPI_COMM_WORLD);
19.    ..................computation
20.  }
21.  ..................computation
22. }
```

While finding communication pattern of a program, the focus should be given to sender, receiver, and size of the message, not on the actual result of program execution. It is sufficient to record the sender, the receiver and the size of the message without executing communication or computing instructions. This can be done by using instrumentation techniques. We have developed the DAPE technique to find communication pattern of

MPI programs and its details are given in section 5. The preprocessed MPI program is given as input to this algorithm to perform instrumentation. In preprocessing the instructions from the input program are organized in such a way that each line will contain single instruction. In the second step, necessary variables and instructions containing those variables are identified. The necessary variables are those variables from MPI communication function which indicates either the sender or the receiver or the size of a message. In the third step, instrumentation instructions are added to generate a trace for communication pattern of MPI programs. During the first run of a program, only instructions containing necessary variables and instrumentation instructions are executed. The instrumented program for the program in listing 1 is shown in listing 2.

Listing 2
The instrumented program for sample MPI program

```
1. int main(int argc, char * argv[])
2. {
3.    int rank, numprocs;
4.    char buffer[20];
5.    MPI_Init(&argc, &argv);
6.    MPI_Comm_size(MPLCOMM_WOLRD, &numprocs);
7.    MPI_Comm_rank(MPLCOMM_WOLRD, &rank);
8.    // ..................computation
9.    if(rank%2==0)
10.    {
11.      //..................computation
12.      MPI_Send(&buffer,10,MPI_CHAR,rank+1,tag,MPLCOMM_WORLD);
13.      fprintf(logfile,"Sender:%d␣Reciever:%d␣Size:%d","src,dest,size");
14.      //..................computation
15.    }
16.   else
17.    {
18.      //..................computation
19.      MPI_Recv(&buffer,10,MPI_CHAR,rank-1,tag,MPLCOMM_WORLD);
20.      fprintf(logfile,"Sender:%d␣Reciever:%d␣Size:%d","src,dest,size");
21.       //..................computation
22.     }
23.   //..................computation
24. }
```

The instrumented program is compiled by using MPI compiler and executed as usual with a required number of processes. Each process will write communication logs in 'logfile'. By merging these log files communication patterns for the entire program is accumulated and communication matrix for a program is generated. That matrix gives topology of MPI program which will be further used for topology-aware process placement.

4.2. Assumptions for partial execution.

Assumption 1: The time taken for execution of MPI program involves computation time and communication time:

$$\text{Execution time of program} = \text{computation time} + \text{communication time} \tag{4.1}$$

In dynamic analysis technique, during the first run of a program both communication and computing instructions are executed. Therefore for execution of large programs, more time will be taken. In DAPE technique, instructions containing necessary variables (variables from MPI communication functions) and instrumentation instructions are executed instead of executing all instruction in the program. This will minimize the time required for program execution. In this way, a program is partially executed instead of complete execution.

Assumption 2: The communication pattern of application depends on two conditions:

F1: Control flow of program
If sending or receiving of a message depends on control instruction in the program, then 'F1' condition will be satisfied.
F2: Data flow in the program
If sending or receiving of a message depends on receiving a message from any process then 'F2' condition will be satisfied.

If the communication depends on F1 then for each and every execution same communication pattern will be generated. But if it depends on F1 and F2 then communication pattern depends on input data and will be different for each execution.

5. Dynamic Analysis with Partial Execution. The 'INSTRUMENTOR' procedure from Algorithm 1 finds message blocks which are executed by different processes of the program. From message blocks, the necessary variables are identified and added to list of the array as shown in 'FINDNECESSARYVARIABLE' procedure. During instrumentation of an MPI program for each line following actions will be taken based on the type of instruction:

I. If the line contains necessary variable then do not put the comment on that line [Line 17].
II. If the line contains control instructions then do not put the comment on that line [Line 18].
III. If the line contains communication instructions then put the comment on that line and also write instrumentation instruction [Line 19].
IV. If the line does not contain necessary variable and it is declaration type then put a comment on that line [Line 20].
V. For any other type of instructions put a comment.

5.1. Parameters for comparing proposed technique. The DAPE technique has the following merits over the existing techniques:

1. Time Efficiency:
In dynamic and symbolic analysis complete execution of a program is needed therefore the time required for analysis is more. In case of DAPE technique very less time is required as the program is partially executed.
2. The accuracy of communication pattern:
Even though DAPE executes program partially, it executes all necessary instructions on which communication pattern depends. Therefore it finds correct communication pattern like dynamic and symbolic analysis does.
3. Scalability:
In dynamic analysis as the number of processes in a program increases the time required for analysis also increases. The DAPE technique takes the same amount of time for any number of processes as it executed only required instructions. Therefore it is scalable with respect to a number of processes.

6. Experimental results. We have conducted experiments to check the accuracy, performance, and scalability of the proposed technique. All the experiments are conducted on WCE-Rock cluster [22]. This cluster contains three nodes connected using InfiniBand network and total 56 cores. Each core has the processing power of 2.25 GHz. The total main memory in the cluster is 29 GB and physical storage of 5 TB. For testing purpose, ten MPI programs having different communication patterns are used. As shown in Figures 6.1(a) and 6.1(b), the program execution traces of DAPE are simple as that of ISP tool. Therefore time required to find communication pattern is less in case of DAPE technique.

6.1. The accuracy of DAPE. The communication patterns for ten MPI programs are found by using both ISP tool and DAPE technique. From each pattern number of point to point and collective calls in the program are found. As shown in Table 6.1, the DAPE technique finds a same number of point to point and collective communication calls as that of ISP tool.

By analyzing the execution traces (Figures 6.1(a) and 6.1(b)) generated by ISP tool and DAPE technique, the communication pattern of MPI program is stored in matrix format as shown in Tables 6.2 and 6.3. To check the accuracy of DAPE technique, communication matrix generated by it is compared with communication

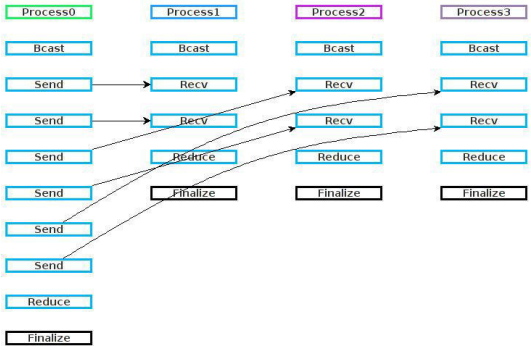
Algorithm 1 DAPE algorithm

```
Input : P                                                    ▷ The MPI program
Output : P'                                                  ▷ Instrumented MPI program
procedure PREPROCESSING(P)
    while (!EOF(P)) do
        L = Read line from P
        if ('L' Contains single instruction ) then           ▷ Skip that line
        else
            Arrange instructions such that each line contains single instruction
        end if
    end while
end procedure
---------------------------------------------------------------------------
procedure INSTRUMENTOR(P)
    NV=FindNecessaryVariables(P)
    while (!EOF(P)) do
        L = Read line from P
        if ('L' Contains Necessary Variable from NV) then    ▷ Don't put comment on that line
        else if ('L' Contains Control Instruction) then      ▷ Don't put comment on that line
        else if ('L' Contains Communication Instruction) then ▷ Perform instrumentation and put a
comment on that line
        else if ('L' Contains Declaration Instruction) then  ▷ Put comment on that line
        end if
    end while
    return P'
end procedure
---------------------------------------------------------------------------
Input : P                                                    ▷ The MPI program
Output : NV                                                  ▷ List of necessary variables from program p
procedure FINDNECESSARYVARIABLES(P)
    Find message block
    Find communication function
    Get 4th parameter from the function. If it is variable add to the list of necessary variable(NV)
    Get 1st and 2nd parameter and compute the size of the message from both parameters
    Get rank variable from control flow instruction & determine the source of message
    return NV
end procedure
```

matrix generated by ISP tool. The DAPE technique has the same accuracy like ISP tool as it generates same communication matrix as ISP tool. Thus the accuracy of DAPE technique is same as that of ISP tool.

6.2. The efficiency of DAPE. The time required for MPI program analysis is used to measure the efficiency of the technique. As shown in Figure 6.2, DAPE takes less time for analysis as compared to ISP tool.

6.3. Scalability of DAPE. To find communication pattern of MPI program for the different number of processes, the program is executed with a different number of processes. In case of ISP tool, as the number of processes in the program increases the analysis time also increases. But in case of DAPE tool, there is less variation in analysis time even if a number of processes increases. As shown in figure 6.3, DAPE is scalable with respect to a number of processes in the program.



(a) Analysis of MPI program using ISP

```
MPI_Bcast():- Sender:0 Receiver:All Message-size=4
MPI_Send():- Sender:0 Receiver:1 Message-size=8 Tag:1
MPI_Send():- Sender:0 Receiver:1 Message-size=8 Tag:2
MPI_Send():- Sender:0 Receiver:2 Message-size=8 Tag:1
MPI_Send():- Sender:0 Receiver:2 Message-size=8 Tag:2
MPI_Send():- Sender:0 Receiver:3 Message-size=8 Tag:1
MPI_Send():- Sender:0 Receiver:3 Message-size=8 Tag:2
MPI_Bcast():- Sender:0 Receiver:All Message-size=4
MPI_Reduce():- Sender:0 Receiver:All Message-size=8
MPI_Bcast():- Sender:0 Receiver:All Message-size=4
MPI_Reduce():- Sender:0 Receiver:All Message-size=8
MPI_Bcast():- Sender:0 Receiver:All Message-size=4
MPI_Reduce():- Sender:0 Receiver:All Message-size=8
MPI_Reduce():- Sender:0 Receiver:All Message-size=8
```

(b) Analysis of MPI program using DAPE

FIG. 6.1. *Analysis of MPI program*

TABLE 6.1
Comparison of DAPE technique with ISP tool [No. of processes=80]

MPI Program	ISP Tool		DAPE Technique	
	#Calls (Point to Point) + (Collective)	Analysis Time	#Calls (Point to Point) + (Collective)	Analysis Time
Array.c	28+1	4.087654	28+1	2.029162
Blocking-Comm.c	1 (deadlock)	2.062398	8+0	0.0068
Hellosend.c	1+0	2.115488	1+0	0.00003
Matrix-mul.c	49+0	4.474906	49+0	0.001052
Quad.c	14+2	3.34783	14+2	0.000156
Ring.c	48+0	9.821415	48+0	0.000088
Gauss-Elimination.c	0+8	4.868085	0+8	3.901431
Integration.c	7+0	2.070263	7+0	0.198444
Safty.c	0+1	2.376012	0+1	0.246936
Prime.c	0+38	11.419859	0+38	9.366117

7. Conclusion. The communication latency is the performance bottleneck for MPI programs on multi and many core systems. By using topology-aware process placement, this latency can be minimized. In literature, various MPI program analysis techniques are proposed. Each technique has its own pros and cons. In this paper, we have proposed a new technique for analysis of MPI program. The DAPE technique finds correct communication pattern in a program except for few programs where conditions F1 and F2 both are satisfied i.e. where program communication pattern depends on both data flow in the program and control flow of the program. The time efficiency of DAPE technique is more as compared to ISP tool. Also, it is scalable with a number of processes in the program as compared to ISP tool.

In future, we are planning to extend the functionality of DAPE which will correctly detect communication pattern of a program even if both F1 and F2 conditions are satisfied. Also, we are planning to integrate our technique with Open MPI so that it can be used while compiling the program.

REFERENCES

[1] MESSAGE PASSING INTERFACE FORUM, *MPI: A message-passing interface standard*, MPI forum., (2009).
[2] GUILLAUME MERCIER AND JEROME CLET-ORTEGA, *Towards an Efficient Process Placement Policy for MPI Applications in Multi-core Environments*, Recent Advances in Parallel Virtual Machine and Message Passing Interface, Volume 5759 (2009), pp. 104-115.

TABLE 6.2
Communication matrix generated by DAPE for 'Ring.c' program

	P0	P1	P2	P3	P4	P5
P0	0	50	0	0	0	0
P1	0	0	50	0	0	0
P2	0	0	0	50	0	0
P3	0	0	0	0	50	0
P4	0	0	0	0	0	50
P5	50	0	0	0	0	0

TABLE 6.3
Communication matrix generated by DAPE for 'Matrix_mul.c' program

	P0	P1	P2	P3	P4	P5
P0	0	4	4	4	4	4
P1	3	0	0	0	0	0
P2	3	0	0	0	0	0
P3	3	0	0	0	0	0
P4	3	0	0	0	0	0
P5	3	0	0	0	0	0

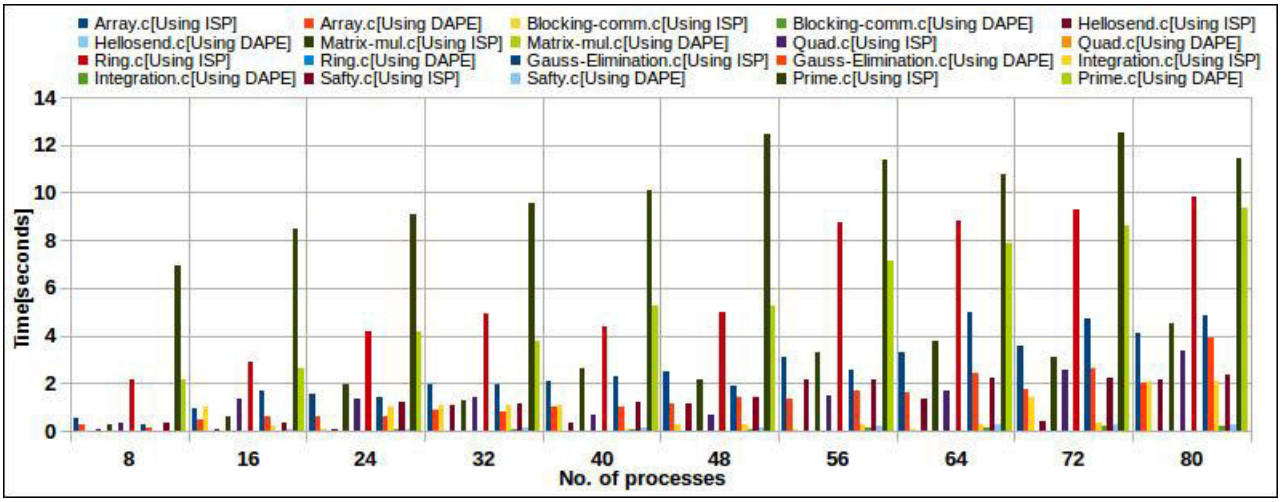


FIG. 6.2. *Time required for analysis: ISP vs DAPE*

[3] S. F. SIEGEL AND G. GOPALAKRISHNAN, *Formal Analysis of Message Passing*, Lecture Notes in Computer Science, volume 6538 (2011), pp.2-18.

[4] G. GOPALAKRISHNAN, ROBERT M. KIRBY, STEPHEN SIEGEL, RAJEEV THAKUR, WILLIAM GROPP, EWING LUSK, BRONIS R. DE SUPINSKI, MARTIN SCHULZ, GREG BRONEVETSKY, *Formal Analysis of MPI-Based Parallel Programs: Present and Future*, ACM transaction on Communication, (2011), http://www.computer.org/portal/web/csdl/doi/10.1109/SC.2010.

[5] SHUYI SHAO, YU ZHANG, ALEX K. JONES, RAMI MELHEM, *Symbolic Expression Analysis for Compiled Communication*, Proceedings on IEEE International Parallel and Distributed Processing Symposium, ISBN: 978-1-4244-1693-6 (2008), pp.1-8.

[6] DALE R. SHIRES AND LORI POLLOCK, *Program Flow Graph Construction for Static Analysis of Explicitly Parallel Message-Passing Programs*, University Of Wisconsin-Madison, (2000).

[7] MICHELLE MILLS STROUT, BARBARA KREASECK, PAUL D. HOVLAND, *Data-Flow Analysis for MPI Programs*, Proceedings on International Conference on Parallel Processing, ISBN: 0-7695-2636-5 (2006), pp.175-184.

[8] ANDREW J. MCPHERSON, VIJAY NAGARAJAN, AND MARCELO CINTRA, *Static Approximation of MPI Communication Graphs for Optimized Process Placement*, The 27th International Workshop on Languages and Compilers for Parallel Computing, (2014).

[9] JULIEN JAEGER, EMMANUELLE SAILLARD, PATRICK CARRIBAULT, DENIS BARTHOU, *Correctness analysis of MPI-3 non blocking communication in PARCOACH*, European MPI Users Group Meeting, Bordeaux, France, (2016).

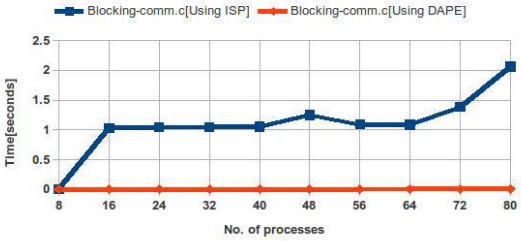
[10] SRIRAM ANANTHAKRISHANAN, GREG BRONEVETSKY, MARK BARANOWSKI, *ParFuse: Parallel and compositional analysis of message passing programs*, Lecture notes in computer science, Springer (2017).

[11] SARVANI S. VAKKALANKA, SUBODH SHARMA, GANESH GOPALAKRISHNAN, ROBERT M. KIRBY, *ISP: a tool for model checking MPI programs*, Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming, ISBN: 978-1-59593-795-7 (1971),pp.285-286.

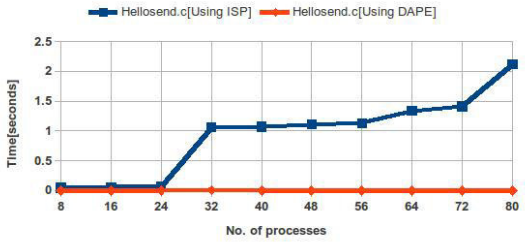
[12] ANH VO, SARVANI VAKKALANKA & RAJEEV THAKUR, *Formal Verification of Practical MPI Programs*, Proceedings of the 14th ACM SIGPLAN symposium on Principles and practice of parallel programming, ISBN: 978-1-60558-397-6 (2009), pp.261-270.

[13] HU CHEN, WENGUANG CHEN, JIAN HUANG, BOB ROBERT, H. KUHN, *MPIPP: an automatic profile-guided parallel process placement toolset for SMP clusters and multiclusters*, Proceedings of the 20th annual international conference on Super-

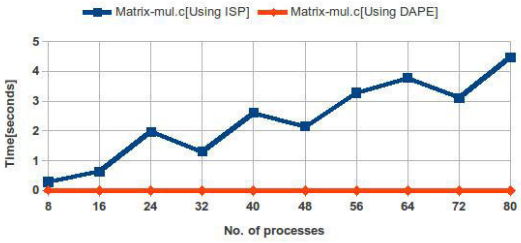
(a) Array.c



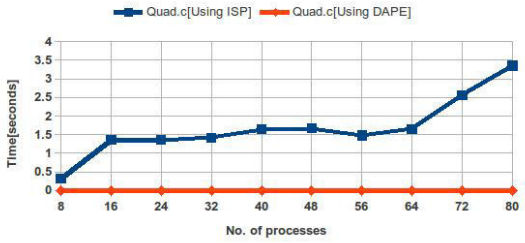
(b) Blocking-comm.c



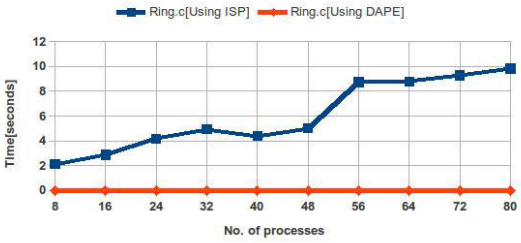
(c) Hellosend.c



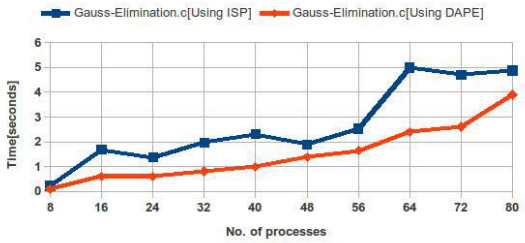
(d) Matrix-mul.c



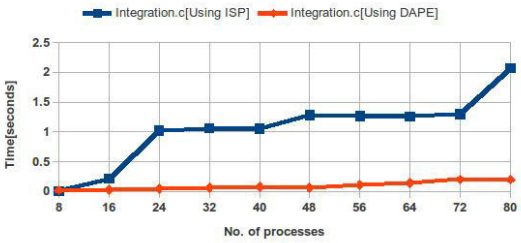
(e) Quad.c



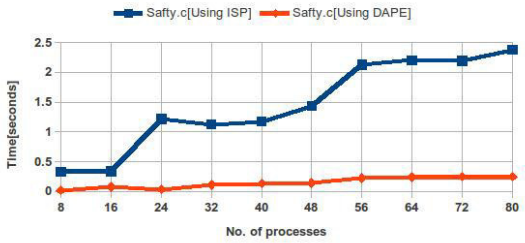
(f) Ring.c



(g) Gauss-Elimination.c



(h) Integration.c



(i) Safty.c

FIG. 6.3. *Scalability of ISP vs DAPE*

computing, ISBN:1-59593-282-8 (2006),pp.353-360.

[14] Jidong Zhai, Tianwei Sheng, Jiangzhou He, Wenguang Chen, Weimin Zheng, *FACT: fast communication trace collection for parallel applications through program slicing*, Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, ISBN: 978-1-60558-744-8 (2009), Article-27.

[15] Yan Li, Jidong Zhai, Keain Li, *Communication analysis and performance prediction of parallel application on large scale machines*, Advances in system analysis, software engineering and high performance computing book series (2016).

[16] Vojtach Iorejt, Saurabh Joshi, Daniel Kroening, Ganesh Narayanaswamy, Subhodh Sharma, *Precise predictive analysis for discovering communication deadlocks in MPI programs*, ACM transaction on programming, languages and systems, Volume 39, issue 4, (2017).

[17] Xianjin Fu, Zhenbang Chen, Yufeng Zhang, Chun Huang, Wei Dong and Ji Wang, *MPISE: Symbolic Execution of MPI Programs*, Proceedings of IEEE 16th International Symposium on High Assurance Systems Engineering, ISBN: 978-1-4799-8111-3 (2015),pp.181-188.

[18] Sultan Aljahdali, Mosaid Al Sadhan, Alaa Ismail Elnashar, *Two automated techniques for analyzing and debugging mpi-based programs*, 24th International Conference on Computers and Their Applications in Industry and Engineering, ISBN: 9781618393227 (2011),pp.344.

[19] Shirley Moore, David Cronk, Kevin London, and Jack Dongarra, *Review of Performance Analysis Tools for MPI Parallel Programs*, Proceedings of the 8th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface, ISBN:3-540-42609-4 (2001), pp.241-248.

[20] Sriram Aananthakrishnan, Ganesh Gopalakrishnan, Bronis R. de Supinski, Martin Schulz, Greg Bronevetsky, *A Scalable and Distributed Dynamic Formal Verifier for MPI Programs*, Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking,Storage and Analysis, ISBN: 978-1-4244-7559-9 (2010), pp.1-10.

[21] Stephen F. Siegel, Yi Wei, Timothy K. Zirkel, *TASS: The Toolkit for Accurate Scientific Software*, Proc. Mathematics in Computer Science, (2011).

[22] http://wce.ac.in/it/landing-page.php?id=9.

Edited by: Dana Petcu
Received: May 19, 2017
Accepted: Nov 13, 2017

AIMS AND SCOPE

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

Expressiveness:

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

System engineering:

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

Performance:

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

Applications:

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

Future:

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

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

INSTRUCTIONS FOR CONTRIBUTORS

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

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

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

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

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