ASSESSMENT OF DATA AND INTERFACING OF SERVICE-ORIENTED SYSTEM WITH SENSORS

Ph.D. Thesis

By

ROBIN SINGH BHADORIA

(Roll No. 1301201005)

DISCIPLINE OF COMPUTER SCIENCE & ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

JANUARY 2018

ASSESSMENT OF DATA AND INTERFACING OF SERVICE-ORIENTED SYSTEM WITH SENSORS

A THESIS

Submitted in partial fulfillment of the requirement for the award of the degree
of
DOCTOR OF PHILOSOPHY

by

ROBIN SINGH BHADORIA
(Roll No. 1301201005)

DISCIPLINE OF COMPUTER SCIENCE & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY INDORE
JANUARY 2018

ACKNOWLEDGEMENTS

First of all, I would like to thank the ultimate source of energy, *the nature*, for blessing and motivate me to pursue my research work. Then, I would like to thank my supervisor, Prof. Narendra S. Chaudhari, for his valuable guidance, encouragement, and direction throughout my thesis work. It is my great honor and pleasure to work under the esteemed guidance of Prof. Narendra S. Chaudhari. I would like to express my gratitude towards my co-supervisor, Dr. Amod C. Umarikar and sincerely thanks for kind leading suggestions to complete my research work.

I would also like to show my heartfelt obligations towards my PSPC committee members, Dr. Aruna Tiwari and Dr. Ram Bilas Pachori for their interesting discussions and suggestions towards my research work. Also, I extends my deep regard to Dr. Kapil Ahuja, for his support and great help during my research work at IIT Indore. I would like to pay my sincere thanks to Head of the Department, Dr. Surya Prakash for his support. Overall, I like to thank to everyone who has helped me to conduct my research at IIT Indore.

I would like to express my gratefulness to Prof. Geetam S. Tomar, Director, Machine Intelligence Research (MIR) Lab, Gwalior, India who had sponsored me for having my PhD Programme at IIT Indore.

Whatever little I have achieved in my life, the credit goes to my lovely family members: my parents, my grandparents, my wife and my brother whom I take great pride in dedicating this thesis. They have always been a source of inspiration for me and have kept trust and faith in whatever I did.

Finally, I would like to thank all the academic and non-academic staff members of IIT Indore who have made my stay easy at IIT Indore.

Robin Singh Bhadoria

Dedicated to My Beloved Family

Contents

Abstract

Data is a key factor in delivering service across the networks. It is an important aspect in handling service interaction which governed by instructions to facilitate legacy application like wireless sensor network. Such data is generally produced in passes and it is used as interaction parameters for the communication between two parties. This integration between multiple service applications could be strengthened by adopting efficient and effective mechanism for these web services. The Enterprise Service Bus (ESB) satisfies all the rules for handling these web services and it also extends the feature for service-oriented architecture (SOA) like routing, protocol transformation, mediation support for existing IT assets and many more.

ESB is middleware framework that helps in designing and developing web services through which software intermediary could be possible. This could also be best solution for handling data from sensors and provide underlying support for protocol conversion between gateway and cloud. ESB is a kind of depletion layer that efficiently handles various overheads during communication. This thesis investigates the detailed issues about ESB and its implication with different mediation solution. This also includes essential design principles, architectural framework, routing functionalities, and survey for existing ESBs available in the market.

This thesis also presents the detailed mathematical formulation that derives the relationship between multiple services (also termed as service components) for handling data through common and shared platform of ESB. Data from various sensors is collected through gateway and it need to check for noise/error before storing such data to global repository. Such noise/error in data may arise due to sudden voltage fluctuation, lose connection, short circuit and link breakage problems. This fluctuation in the data value creates unstable, discontinuity and noisy and need to be stabilize before storing.

This thesis advises an “Intelligent Sensor Network” (*IntSeN*) methodology which is multi-level architecture for handling sensor data and filter data on the basis of delay, density, type and value threshold. Sensed data may contain noise or error due to environmental hindrances or voltage fluctuations that must be reduced for stabilizing an overall system. Such methodology reduces the relatively error by more than 50 % that occurred during sensing process.

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

XML: Extensible Markup Language

JMS: Java Messaging Service

TIBCO: The Information Bus Company

EAI: Enterprise Application

JMX: Java Management Extension

WSN: Wireless Sensor Network

SOA: Service-Oriented Architecture

ESB: Enterprise Service Bus

J2EE Connector Architecture (JCA)

J2EE: Java to Enterprise Edition

HTTP: Hyper Text Transport Protocol

SOAP: Simple Access Object Protocol

FTP: File Transfer Protocol

QoS: Quality of Service

EIP: Enterprise Integration Pattern

SE: Service Engine

IntSeN: Intelligent Sensor Network

BPM: Business Process Management

JDBC: Java Data Base Connector

OGC: Open Geospatial Consortium

SWE: Sensor Web Enablement

SDK: software Development Kit

IoT: Internet of Things

BPMS: Business Process Management System

DoS: Denial of Service

SLA: Service Level Agreement

B2B: Business to Business

Chapter 1

Introduction

In an era of rapidly changing technology, there is a demand for service integration of different application domains that share various IT assets within and outside organization. A consequence of this change in demand is highlighted in [3, 22] which are as follows:

- If an organization expands its business, then it requires restriction-free environment for its business.
- Changing demand of market needs elastic, scalable and compliant system architecture.
- Complete utilization of existing IT assets.
- Compliance with maintenance, reconfiguration, and reusability of system services.

In the context of issues highlighted above, a service-oriented system would be an appropriated solution for layered architecture to service integration, reusability, and scalability to multiple services. Due to the distributed behavior of such systems, it has no region boundaries i.e. services from different domains can interact with each other independently. It also supports multiple data formats that help in service interaction and increases the agility and accessibility for the overall system [18]. Such systems could dynamically handle simultaneous requests from different clients through multiple interfaces (provided endpoints). It actually supports all the aspects of Service-Oriented Architecture (SOA) that includes resource sharing, message handling, translation, data transformation, protocol conversion, service versioning, and adopting all service integration methodologies expressed in SOA [2, 5].

1.1 Overview for Service-Oriented Systems

Service-orientation is a methodology to software system that is prevalent fashion for development and deployment of web services. It offers the features like platform independence, standardization of protocol, well-defined interfaces, and support loosely coupled services. The primary motives for adopting service orientation are interoperability and reusability. Such systems require a lot of effort in processing and handling somewhat more implementation ethics than simple SOA based systems. This introduces a new concept called - '*Integration Suite*' that provides extension to the features of SOA and explores architectural benefits of distributed and flexible framework. This framework is most adopted by Enterprise Service Bus (ESB) that works in distributed kind of manner to facilitated services across the network. ESB sometimes work like a central hub, but with distributed agent. These features make ESB as more demanding and adorable service-oriented solution for SOA systems [48].

The word "Enterprise Service Bus" was first introduced by Roy W. Schulte from Gartner Group in 2002. In June 2004, David Chappell wrote a book -*The Enterprise Service Bus*, which briefs about the several aspects for service-oriented based systems. In 1987 *Teknekron Corporation, USA* project with Goldman Sachs first introduced "*The Information Bus (TIB)*", which defined integration and delivery of market information like stock quotes, news and other financial figures.

In 1997, Vivek Ranadivé founded a company named TIBCO (stands for ***T**he **I**nformation **B**us **CO**mpany*), which provided service, especially, other than financial sector and released TIBCO Active Enterprise suite. But there is a great demand in the industry as well as business for acceptance of ESB with SOA system as new challenges [16, 35].

1.1.1 Opportunity with ESB

ESB is best in-class to integrate, develop, deploy and manage multiple services on the common platform. ESB is ordered to perform and deploy services with SOA technology with followings:

- To moderate cost issues
- To speed-up IT implementations
- To reduce IT complexity

Organizations can align its business with the use of appropriate ESB selection in a specific domain of services and reuse multiple applications through web services [18]. ESB also enables developers to provide a rich, integrated and interactive development environment to build federated services. This inclusion of ESB improves the rapid development of integration logic, which does not require detailed knowledge about J2EE or Java. ESB supports virtualization and handling features that execute and extend the core functionalities of SOA. ESB could be adopted due to following reasons:

- Easy-to-use mediation proficiency
- Extensive connectivity with existing IT assets
- Integrated framework for development & deployment
- Improve manageability via single point of control
- Better flexibility to its existing systems
- Ability to extend user's environment
- Improved performance & optimized networks
- Maximized business ability & adaptations
- Secure and dynamic routing for its messages
- API adaptable to know enterprise pattern
- Implement and execute the virtualization without revealing identity & location

ESB is typical infrastructure in software engineering area. It provides the support for services, security handling, and monitoring the quality of service. It has been analyzed that concept of ESB is to connect various service components together i.e. service containers. It is a good option for building platform in enterprise integration, as it protects and hides transport protocols information. It has the capability to transport messages or data more securely. It also used to integrate several services from hybrid and homogeneous systems as a common solution for middleware. ESB also improves the architectural benefit associated with an enterprise solution for service integration and well-organized service handling. By adopting ESB as their business solutions, one can reduce the risk that involved the data management issues and resultant would automatically increase the Return on Investment (RoI).

1.1.2 ESB as Vibrant Middleware Solution

ESB is middleware mediator that supports transport protocol conversion, message content, and format between multiple services from the different domain of applications. Web service actually deployed through service components, called service containers, which utilize the resource instance. The ESB framework is designed to transmit information and routing between multiple service applications. The idea of ESB is related to; Enterprise Application Integration (EAI) and Service Oriented Architecture (SOA). It also envisioned the perception of service integration as federated solution as shown in Figure 1.1. In traditional practices, service integration is achieved by point to point reconciliation. The individual platform is designed to support each one of new application with existing platform. This is an extremely clumsy methodology and has to be replaced with the federated solution of service orchestration on a common and shared platform. Service container communicates with each other by sending a message and these messages need to be routed through the well-known path. ESB is efficient in delivering such messages through well-known and dynamic routing algorithms. ESB also act as message-based distributed integration middleware that provides interaction in distributed manner and builds a sophisticated, secure and reliable meditation [54].

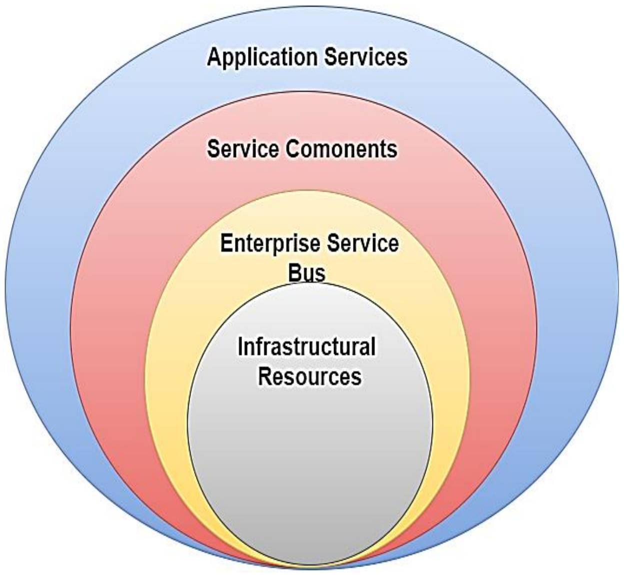


Figure 1.1: Positioning of ESB in Relation to Service Components and Infrastructural Resources

ESB is designed to nurture integration methodology and handles resource provisioning between multiple service containers by supporting web services communication protocols. These containers are usually dynamic and plug-in to ESB. The self-measurement principle of ESB in monitoring and diagnose concurrent system resources would make more adorable efficiency in delivering services across networks. This efficiency could depend on multiple parameters that may influence the performance of any SOA based system. These parameters must be diagnosed and formed Quality of Service (QoS) on which any ESB may rely. This must understand the dynamic need of any enterprise application and have to be adaptable with ESB. It means that applications have to pursue some rule to adapt to these new necessities. It should possess the QoS-aware monitoring of different parameters depending on the particular application [57].

ESB supports data formats of the specific message and these formats are provided by each ESB manufacturer via Enterprise Integration Patterns (EIP). During the communication between multiple services from different domains through messaging, ESB helps in recognizing these message formats through known EIPs. ESB also provide appropriated path to route these messages to avail right service [12].

1.1.3 Foundation Principle for ESB

There are a several unique perspectives on what core functionalities for an ESB depends on. The rundown on which ESB performance and evaluation reply are important. Service provider plays a vital role in delivering these functionalities and must be committed in its service client as shown in Figure 1.2. A message itself is a piece of data or some kind of data structure that contains the information related to string, byte array, a record or an object. The message specifies the event occurrence between its sender and corresponding receiver that command message and document message [11, 14].

a. **Transport protocol conversion:** ESB helps in adaptability and understanding the underlying communication protocol that governs by that particular application service. This feature of ESB provides the connectivity with old legacy systems to new modern computing systems [7].

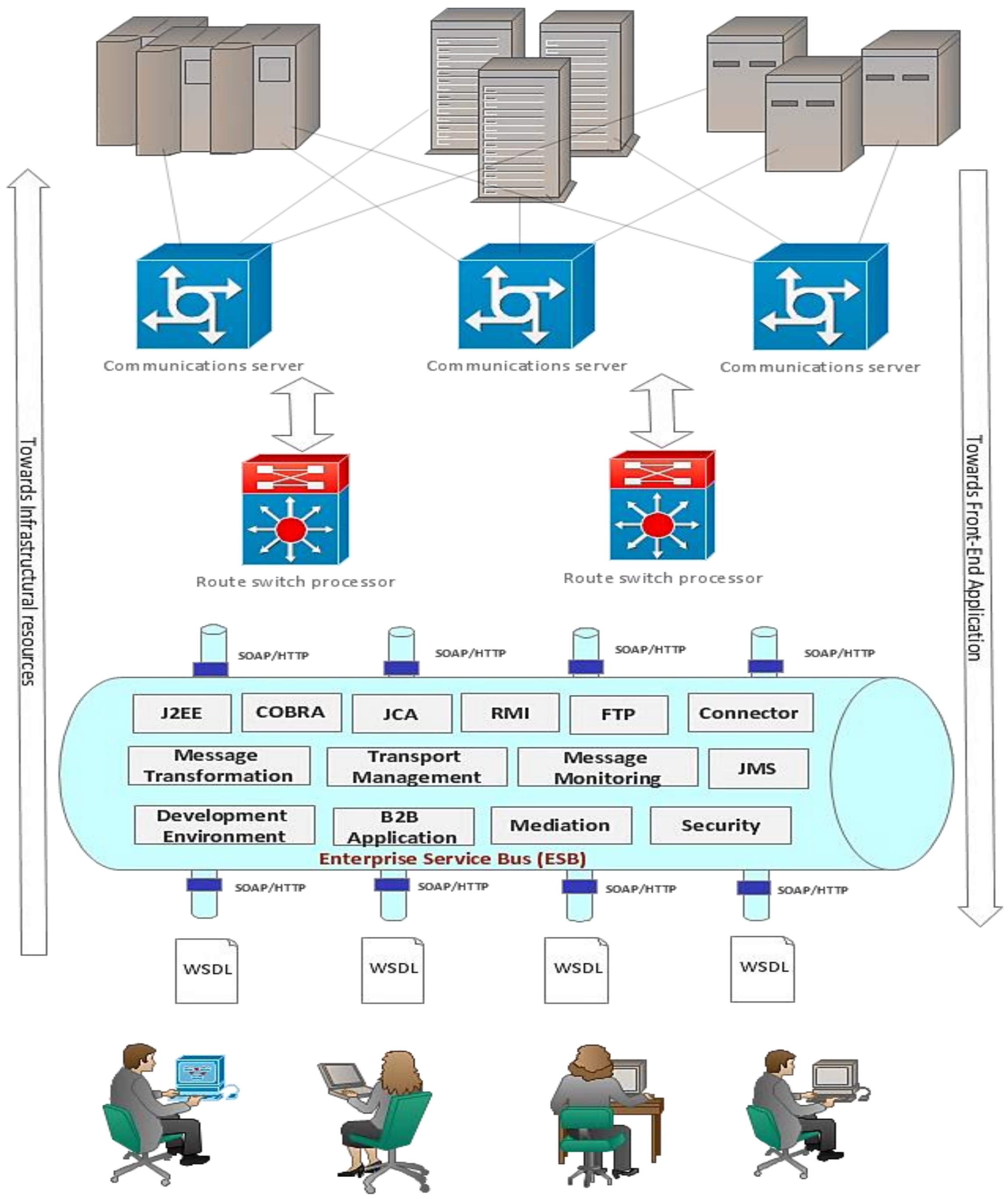


Figure 1.2: Relationship of Infrastructural Resources and Web Service Consumers with ESB as middleware

b. **Mediation and Message transformation:** This functionality provides translate of message format between communication service components that is necessary to deliver federated service of different systems on a common platform. This also supports inbound and outbound protocol transformation [4].

c. **Message routing:** This is one of the most striking feature in ESB that changes the mood of market and business to shift their commerce through ESB. It enables the service provider to send its message to designated path [13].

d. **Message enhancement:** This feature enables the service provider to put additional or missing information of data needed at the time of communication. It not only improves the communication overheads, but also categorizes demand needed at the time of resource access. This could best be possible with help of EIPs that have been provided by every ESB manufacturer [58].

e. **Security Aspect:** With this functionality in ESB, it could enable the service providers to handle different security parameters which are directly associated with validation an authentication check. It also verifies the web service for its integrity, confidentiality, and availability through WS-security protocol. The information associated with the interaction of multiple services are protected bound so that any malicious party would not able to access such information [60].

f. **Monitoring and management:** This feature enables the service provider to improve the efficiency and flexibility by providing monitoring tool via ESB console (Console is IDE which is different for different ESB). This is best supported by Java

Management Extension (JMX). There are different monitoring tool that supported by particular ESB which could detect mediation and message flow along with the trace of client patterns in the desired transaction [6].

g. **Direct proxy:** This feature has the capability to transfer received messages into specific type service that could help in building federated service. Several ESB products have been fitted with service proxy capabilities. Such capabilities make easier to integrate (federate) multiple systems in SOA [59].

It is important to note that ESB manufacturer should provide guidelines in the form of policy that must follow by their clients/customers for best manageability, accessibility, and observability. However, such policies need large consideration of ESB manufacturer that governs the data and facts which are propagated from lower layer to higher layer of SOA reference architecture [61]. Several experts are doing their research to find the quality perspective for ESB and could benefit society in best available ESBs selection (commercial as well as open-source). Determining the service interaction methodology is a significant architectural decision for experts as well as researcher. It could affect the overall performance for implementing web service across networks [52]. So, simulation of these available ESBs is important to segregate QoS based parameter which includes: received and sent messages, time of execution, overall bytes utilized, message counts, memory utilization etc.

1.2 Wireless Sensor Networks and its feasibility

Wireless Sensor Network (WSN) is an autonomous collection of devices that are interconnected via wireless media. It comprises various spatially distributed devices which monitor the environmental parameters like temperature, light, humidity, and various oxide gases. Such WSN system includes '*a gateway*' that provides a platform for common sharing and forwarding captured information or data. This scenario is well depicted in Figure 1.3. The governing protocols and standards for such sensor-based networks depends on the application that is required for implementing the system and it may include 2.4 GHz radios based transreceiver (IEEE 802.15.4 or IEEE 802.11) that usually follow 900 MHz band frequency.

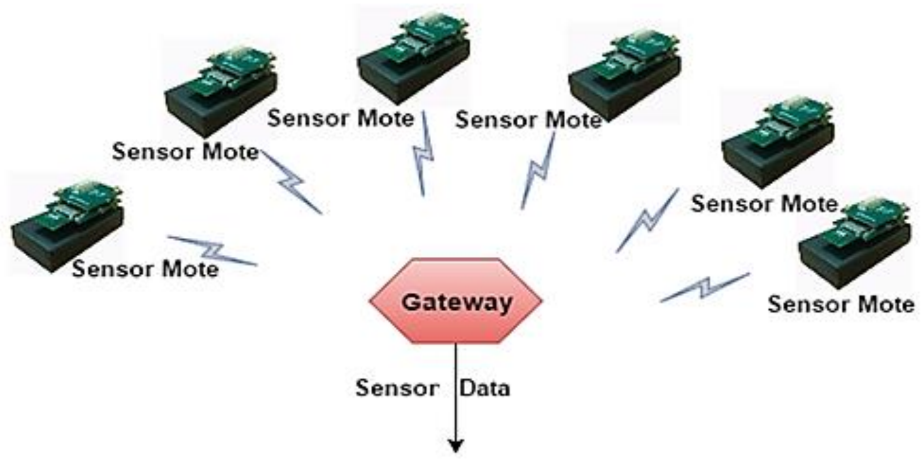


Figure 1.3: Scenario for Wireless Sensor Networks

All captured data is being forwarded through the gateway to destinated repository. Such repositories could be local as well global where all captured/sensed data is kept. So, the role of the gateway as an interface, is important in terms of handling data in more sophisticated manner. Gateways are designed to support flexibility and reusability of services and enable ESB to unleash with business processes migration. It also supports in delivering highly integrated infrastructure for controlling this deployment environmental parameters as shown in Figure 1.3. Such gateway could support in data streaming applications like video data streaming.

1.2.1 Evolution of Sensor Data over the Web

At NASA laboratory, the conceptualization of Sensor data over the Web started in the year 1997 when Kevin Delin first coined this perception. He has also defined a wireless system as distributed, intra-connected, and communicable sensor clusters. These sensors are also installed to capture various events for environmental features like temperature, pressure, light etc. Kevin also pointed out the significance of these systems as “macro-instrument for synchronized sensing”. This project was successful in implementing monitoring of the environment. This idea was implemented in NASA’s Jet Propulsion Labs that could help in the battlefield surveillance purpose and later on, this concept was conceived by several applications. Thus, this integration of sensor with the web gives new hope to sensor network system which not only making sensor device intelligent but also enables the system to establish a connection with a neighboring device for exchanging data. Such system potentially shared the relevant information with cluster sensor device over data acquisition. This acquired information is collected and could be accessible by this web integration. The integration of two different technologies could make the system more sophisticated and efficient in delivering services across the network. This could be achieved by mean of competent middleware technology which supports the sensor data more conveniently over the web. The efficient way of handling sensor data could explore growing possibilities for our technology researcher and practitioner as discussed in [96].

For improvement in sensor and the web integration, Open Geospatial Consortium (OGC) was formed in 2001 that does the specification and open standards development for sensor web integration. This organization also supports rendering the geospatial services which significantly developed the standardization for a sensor device in combination with the web. OGC worked for recognition of sensor networks over the web and made an active and special working group that does standardization of sensor web interoperability issues. This working group is named as Sensor Web Enablement (SWE). It deals with open specifications and characterization of the sensor network with web services. This SWE working group primarily dedicated for standardization for sensor data discovery, acquisition, monitoring, and event recognition. These features are well nurtured and presented in Service-Oriented Architecture (SOA) in which data is modeled using open standards and specifications. It also involves the data encodings that describe the capabilities which include data types, its mode, and phase of activation using web services.

1.2.2 Sensor Data Processing over the web

In environmental monitoring, sensor plays an important role in capturing and sensing the different parameters. It could be possible by using the different variations of sensors which scattered in different geospatial locations. This captured information is stored with help of "data logger" which is a hardware device that accountable for receiving sensed data (raw form) from proximity sensors. This involves the interchange data using analog to digital converter. Such inter-conversion of data is recorded into data logger on local memory until it is forwarded to a central computational unit through the well-defined gateway. The manufacturing company for such data logger offers enterprise built-in applications software (SDK) to monitor the captured data at a central computational unit. This information could be beneficial in controlling and managing deployed sensor devices at different geospatial locations. With the help of these Software development kits (SDKs), clients could also access and execute sensor data information at their end. It could be available via the web interface which could be customized according to client's needs. The collected raw data is analyzed and pre-processed using data cleaning techniques before making it available to client access. This information is extracted from a large set of collected and sensed data. This feature would rely on the monitoring service that assures the quality control for sensed data. Such cleaned data could be available to the data science research community group for developing standardized and open source specification in sensor web enablement [66].

Once data is stored in its corresponding repository (cloud storage), it must be recognized by its data format and style. This could be more specified by the "Not only SQL (NoSQL)" conceptualization in which data could be in structured, semi-structured and unstructured format or structure.

1.3 Research Problem Description

The more realistic problem associated with assessment of data in legacy systems like wireless sensor network. The data created by machine devices is increasing day by day. A large number of embedded devices have been deployed altogether to monitor environmental parameters. Though, such monitoring capabilities are affected by fundamental necessities to mollify specific needs. These necessities are depending on scattered devices (sensor motes) at different geographical locations [114]. The major hurdle in implementing these environmental parameters is interoperability between sensors and client data format which is usually executed by licensed software (for sensor devices). Such data format and style have a specific privilege of access for sensor data via well-designed data portal. Limited access to these datasets make difficulties in analyzes. Researcher/experts could not able to determine significant relationships. To lean this gap of accessibility in the dataset, systems must be designed in such manner that it assists the sensor data in an organized way by adopting open source software. These open-source based software help in recognizing data patterns and formats. These data patterns help in service integration of different domain of applications and resolve the issue of interoperability. It also provides the information that could be useful to business enterprise, research organization, and policy makers for any particular field of industry [66].

This interoperability has a vital role in playing dynamic data acquisition from sensor motes. It enables the system with the capability to exchange set of data or information. Such information could be benefited for protocol transformation and inter-information exchange with efficient use of middleware technology. This middleware technology not only rendered its information into classified manner but also assured its delivery to the corresponding client. The absence of interoperability in sensor applications could mislead the system efficiency and limits its access to real-time simulation. Hence, there has to be some common and share platform which provides a mediator to handle all these data management. Middleware technology leads this gap and support interoperability by providing known set of integration patterns that help in recognizing different data style and structure. This is imperative to note that the sensor community could not be away from the use of ambient middleware technology that provides assured delivery and capture of sensor data across the network. These systems must be designed with standardized mediation technology that makes sensor network's data more sophisticated and accessible.

Secondly, the most important problem is of noise/error in the data received during sensing the environmental parameters. This may cause the interruptions in the communication while storing the data at the desired repository (cloud servers). As cloud storage is '*pay-per-use*' policy and governs the charges for its storage, so one needs to keep non-redundant, clean and filtered data over the cloud. This would save the charge for extra storage as well as cleaned data could assess for a specific application.

The overall goal for this thesis is to lean the gap between handling, storage, and filtration of captured data from different sensor nodes. This would also offer better and more sophisticated organization of data.

1.4 Motivation

The captured data from different sensors is a key factor in delivering services across the network. The sensed data need to be forwarded to the global repository like cloud where it could be accessed by different users. Secondly, noise/error in such captured sensor data occurred while sensing of an environmental parameter and this cause due to hindrances or fluctuation in the sensing mode. If such data is stored at repository where duplicate and noisy data is kept, then it may create a situation of malfunctioning and wrong information prediction. Basically, such repository is kept at the global level at some cloud storage and it needs clean and stabilize data which is free from all hindrances or noise in the data. As cloud storage is ‘*pay-per-use*’ policy and governs the charges for its storage, so it needs to keep non-redundant, clean and filtered data over cloud. This would save the charge for extra storage as well as cleaned data could assess for a specific application. This gap between handling, storage, and filtration onto captured data from sensor networks motive us to investigate and analyze the performance metric on mentioned title.

This sensed data is generally passed to the global repository and before restoring it at the cloud, such data need to be cleaned and filtered. It needs serious monitoring and assessment of communication between sensor motes (devices) that involves the integration of multiple services for handling and monitoring of such data. It also demands sophisticated system that strengthens this overall process and guarantees its successful delivery to associated client side where client actually rises a request for its access. This situation could be best controlled by adopting ESB which is a middleware framework that helps in designing and developing web services. It is a kind of depletion layer that efficiently handles various communication overheads and interaction between multiple services. The specific business applications which are complex, dynamic and highly computational, requires a lot of effort in processing and somewhat more implementation than simple SOA. ESB introduces a new concept called - '*Integration Suite*' that provides various integration methodologies which would be discussed in detail in Chapter 3. The need for sensor web interfacing could be modeled by this ESB for effective routing and protocol conversion of captured data from different sensor motes. This would improve scalability, usability, a flexibility of sensor data in multiple stages where sensed data need to take care before storing to cloud.

1.5 Organization of Thesis

The rest of the thesis is organized as follows: Chapter 2 briefs about the background and related work in regards to enterprise computation and its association with wireless sensor network (WSN). It also detailed analysis and comparison of existing ESB products in the market from companies like IBM, Microsoft, Fiorano, Oracle service bus, WSO2, Sonic ESB. Chapter 3 explains about architecture framework for ESB with its core principle for designing. Further, it highlights the importance of service stack with regard to delivering services in ESB. It also presents a comprehensive architecture view for ESB and what ESB actually comprises. It illustrates the concept of message routing and mechanisms associated with message transformation. It also presents a brief discussion on the classification of routing techniques.

Chapter 4 describes the importance of an architectural semantic for service-oriented computing that starts with the characteristics of software systems that have been recognized with sharing and utilization of resources. Chapter 5 gives the insight for integration of wireless sensor network and enterprise service bus. In this chapter, we proposes "*an intelligent sensor network (IntSeN)*" prototype which is multi-level for handling information from sensor motes and making the overall system as stable by removing noise or errors from sensed data. Lastly, Chapter 6 concludes the thesis with the finding of this research work and the future scope.

Chapter 2

Review and Literature Survey

Recent years have noticed with fast change in the market for ESB technology and its associated integration software. The business organizations are choosing mediation support for their architecture enhancement [4]. ESB as a mediator agent supports in interconnectivity for multiple services i.e. *Service Orchestration*. Such communication is usually based on exchanging messages between the two parties which allows an ESB to receive messages from different clients as a request. This request could be processed in accordance with designated service point (usually end-point in ESB) to facilitate service integration. ESB provides varieties of interfaces or endpoints for different channels to support interaction between multiple services that incorporates several types of IT resources as discussed in [2, 17].

2.1 ESB Features & Analogy

ESB can be termed as a middleware that supports the integration and communication between multiple services. To deliver, ease and fast services through ESB, software service quality and features must be rapid enough to handle the demand of business. Open source, agility, and scalability are major challenges in setting up the enterprise software needs. To gear up with these challenges, one must be committed to open standards and interoperability to proprietary solutions. Open source software could be a better solution to avoid the vendors' features lock-in and temporary availability. It also helps in resolving the issue for application integration and support more sophisticated and fast deployable paradigm over the different range of environments.

Table 2.1 presents the different work done by different authors in analyzing ESB. Initially, ESB framework used as Enterprise Application Integration (EAI) and that grows towards the large scale integration as discussed in [15]. Such integration also supports with other application domains like Persistent Computing System (PCS), Multi-Agent System (MAS), and Heterogeneous Systems etc. This framework could federate lightweight client services with heavyweight computing services on the common platform. Service orchestration could be possible in such environment with RESTful gateway [42]. Knowledge management techniques like ontology had been adopted in enterprise solutions in [44].

Table 2.1: Shows Monitoring Framework, comparative issue and Integration of SOA with ESB

Reference	Monitoring Framework	Comparative Point	Integration with SOA	Comment
Sixto Ortiz Jr. (2007)	Talk about basic building structure for ESB as EAI and its origin in mid-1980's	Does not made any comparison, just presented roadblock for service bus	Show importance of SOA and its integration with standardized web-services like XML for data description; HTTP for message transfer; SOAP for message exchange; WSDL for service description and UDDI protocol for discovering & publishing services	Talk about advantage of ESB over application integration approach
Jiangiang Hu et. al. (2008)	Does not talk about Monitoring framework in ESB	No Comparison	Discuss about "ESBsoa" Methodology, but not unified adapter mechanism	Also restrict availability & Scalability
Deng Bo et. al. (2008)	Explain Event driven architecture for Complex Event Processing	Service selection made on basis of greatest "benefit/Cost" ratio	Discuss ESB Technology with SOA and event driven architecture	Proposed algorithm for event selection, aggregation and Greedy service
Jianwei Yin et al. (2009)	Present "JTangSynergy" which is dependable ESB framework	Design system architecture for Chinese Healthcare Service Integration	Focusing on Service Integration using pluggable platform	This proposed system support service recovery by dynamic service

				replacement module (DSRM)
Gang Li et. al. (2012)	Discuss Hub Model in EAI for large scale integration	Compare ESB functionality with Soft System Bus, which is main component of Persistent Computing System (PCS)	Present ESB as infrastructure for integrated and flexible end-to-end SOA	Present ESB as Message Passing Engine
MarekPsiuk et. al. (2012)	Introduced ESB Meta Model (EMM) with monitoring framework architecture	Comparison made on message model, topology model and measuring task	Discuss ESB based framework for SOA system	This proposed model gather data related to ESB entities like Message transmission, Topology discovery mechanism, Message flow etc.
Keshi et. al. (2014)	Integrated framework based on Ontology and ESB	Build model for Heterogeneous Systems	Implement SOA based ESB for enterprise knowledge management	Discuss about Ontology Structure of System Integration
OndřejHarcuba et. al. (2015)	Presents framework that integrate light weight client with heavy weight ESB	Introduced "REST based ESB gateway (REG)"	Extends the idea of integrating Multi Agent System (MAS) and SOA using RESTful service architecture	Discuss Ontology-based model for semantic representation for information

Table 2.2 shows the comparative analysis on how different authors have discussed different perspectives of capacity, planning and experimental simulation in ESB. This table points out the keen issues related to high-performance computing techniques. Testing is a very crucial issue in ESB that could be used in the phase of capacity planning as discussed in [3]. Event driven architecture could also be combined with SOA system via ESB. Relational algebra could improve efficiencies in an event processing based system in [26]. Scalability and load handling issues could be performed with direct proxy, content proxy and transformation proxy. This experiments on three ESB products- Mule, WSO2, and Apache

ServiceMix in [19]. Parallel application bus could be introduced with ESB based architecture to improve multiple services processing as discussed in [43].

Table 2.2: Shows Capacity, Planning and Testing with Existing ESB

Reference	Capacity & Testing	Simulation	High Computing Techniques	Comment
Ken Ueno & Michiaki Tatsubori (2006)	Test on different ESB related to costing, mediation and performance issues	Experiment for web services with no ESB, ESB Without mediation, ESB with mediation	Discuss about Ultra-light service and capacity testing techniques under high load to determine its maximum performance	Good Analytics for Planning & Testing issues
Deng Bo et. al. (2008)	Test and selection that service yield higher benefit per cost ratio	Experiment on service instance for all associative event processing services	Absence of Relational algebra based event stream processing engine in existing ESB	Presented good analysis for complex event processing system
Sanjay P. Ahuja et. al. (2011)	Test scalability & Load handling with Direct Proxy, Content Based Routing Proxy and Transformation Routing Proxy Scenario	Simulated results on ESB like - Mule, WSO2 and Apache ServiceMix	Explain core functionality of ESB w.r.t. virtualization, mediation and content based routing.	Also, presented subjective assessment on parameter like API, Installation, Ease of development, online support for these 3 ESBs
MarekPsiuk et. al. (2012)	Involves testing topologies that support in service containers using JBI different versions	Experiment made on ServiceMix, OpenESB and JSR-208 ESB products	Implemented in Java Business Integration (JBI) that standardized ESB Patterns	Discuss Aspect Oriented Programming (AOP) that support modularity &service orchestration
RidhaBenosman et. al. (2013)	Demonstrate Massively Parallel Application Bus (MPAB), which is ESB oriented architecture	Discuss operation mode of P-Instructions	Shows the functionality of Multiplexing for massively parallel processing of distributed and delay-insensitive application	This could be future solution for complex processing like core banking, Big data simulation etc.

Table 2.3 illustrates the security aspect of services arises with ESB. Also, justifies the protocol specification that is related with WS-security based authentication, service level agreement (SLA) and encryption. It also discusses the error and logging report presented with architecture for SOA [31]. Presented service bus security as an essential factor of concern into ESB architecture and calculate the error for performance modeling and its validation. This experiment has done on BEA AquaLogic ESB product [10]. ESB could extend the functionality of window communication foundation to offer multiple end-points for published services. Propose the architecture that leverages SOA methodologies in grid computing [7].The error handling for a message along with architecture for EAI is well discussed in [12, 21].

Table 2.3: Presents Security, Policy, audit and logging facility for ESB

Reference	Security	Driven Policy	Preventions	Comment
Jiang Ji-chen et. al. (2006)	Present Enterprise Security concept using Single-Sign-On (SSO), Lightweight Directory Access Protocol (LDAP) Integration	Discuss idea of digital signature, WS-security based Authentication and encryption in ESB	Establish SLA / SLO based on event and performance	Brief about Architecture for ESB
Yan Liu et. al. (2007)	Provide Service Bus Security as High level ESB Architecture	Discuss Service Level Agreement (SLA) alerts including message security, authorization, authentication	Calculates error of performance modeling and validation	Present case study for Loan Application use Mule, BEA AquaLogic Service Bus (ASB)
Jiangiang Hu et. al. (2008)	Discuss SOAP security framework in business Integration	Discuss Policy and its definitions in service adaptation at service layer	Define security authentication at – Channel Layer, Service Layer of SOA based EAI environment	Discuss transport adaption, service adaption and common services in SOA based ESB
Alaa M. Riad et. al. (2010)	Explained security aspects for SOA based grid computing	HTTP transport and WS- specifications, Proposed SOA based Grid computing	*ReliableSession* and *TransactionFlow* protocols	Discuss Windows Communication Foundation (WCF)as a message-based architecture
Jieming	Define Security as core	Use open standards	Discuss 02 models -	Present ESB as

Wu and Xiaoli Tao (2010)	functionality of ESB	like HTTP(S) to JMS, FTP to a file batch, and SMTP to TCP	Java Authentication and Authorization Service(JAAS)and Acegi Security	validate connective Route Transform

Table 2.4 demonstrates the routing aspect that had been covered by several authors. Every business organization that deals in ESB products provide its own API called as enterprise integration patterns (EIP). This EIP supports in recognizing message format from other application service domains. Routing is a core feature for ESB that could propagate the message to its destination. To federate routing, MOM (message oriented model) is introduced that is based on event-driven architecture (EDA) and SOA [16]. Different Dynamic Reliable Service Routing methodologies have been proposed by different authors like - Dynamic Routing in Enterprise Service Bus (DRESR), Efficient & Reliable Dynamic Service Routing (ERDSR) [23, 24, and 25].

Table 2.4: Routing, Protocol Transformation and Pattern Recognition in ESB

Reference	**Service Routing**	**Enterprise Pattern**	**Message Transformation**	**Comment**
Martin Breest (2007)	Discuss message routing facility via MOM and its techniques- itinerary based routing, and content based routing	Promise to construct robust and scalable SOA with MOM	Introduced "Message Oriented Middleware (MOM)" that act as mediation	Brief about service container and discussed how ESB endpoint is similar to servlet in J2EE; service orchestration using BPEL
XiaoyingBai et al. (2007)	Proposed Dynamic Reconfigurable ESB Service Routing (DRESR)	Includes Instance Routing Path (IRP) with combination of Abstract Routing Table (ART) and Abstract Rout Path (ARP)	Routing strengthen and assured message delivery and help in protocol conversion. Implement D-Mule, open source ESB	Does runtime service testing by dynamic service selection
Bin Wu et.al. (2008)	Improved Dynamic Reliable Service	Explained 04 routing	Improved Message routing	Also, discuss important reliable

	Routing and extended DRESR	pattern – Enricher, Static Recipient List (SRL), Message Aggregator (MA), Wire Tap (WT)	and its conversion with Dynamic WT	message routing specifications
Sanjay P. Ahuja et. al. (2011)	Work on Content based Routing with scenario to Direct Proxy, Content Based Proxy and Transformation Proxy	Discuss importance of EAI in ESB	Explained Message transformation as core functionality in ESB	Done statistical analysis for calculating throughput or mean response time
PengXu et. al. (2013)	Proposed Efficient & Reliable Dynamic Service Routing (ERDSR)	Presents ERDSR Pattern to improve efficiency in ESB message routing	Does message processing or transformation	Made comparison for DRESR, DRSR and ERDSR

2.2 Comparison of Exiting ESBs

The available set of ESBs in the market supports SOA platform for its execution. Every enterprise organization provides set of enterprise integration patterns (EIP) against cross-platform support [50]. These patterns help in determining message format and related fields in services. ESB also supports the fundamental features like message routing, transformation, protocol conversion and support for the dynamic environment. This chapter discusses the different ESB products that are available in the market.

WSO2 ESB is an open source ESB that provides better compatibility and high performance in terms of service accessibility. This may be the finest option in the available class of ESBs as it supports almost all features of ESB like - Service governance, monitoring and management, virtual environment and so on. According to eBay Inc., for online shopping, it uses WSO2 Enterprise Service Bus for executing and processing transactions, which is more than 1 billion per day. However, it fails to provide proficient security metrics for authentication [39].

Talend ***Enterprise ESB** is a reliable, scalable and secure choice for service bus for any organization. It is a non-open source and high-performance ESB solution and supports the deployment of services in both on-premises and over the cloud. But, it is not lightweight service bus and this is the reason why Talend* Enterprise ESB does not perform well in large-scale operational batch processing [46].

AdroitLogic UltraESB is lightweight service bus, which is capable of supporting different transport and message formats. It is the first ESB, which supports "Zero-Copy proxy" messaging with non-blocking input/output capabilities. It has the ability to dynamically *load or reload* deployment service units. But, it does not support service composition well. It used WS-Security for configuring service to limited extents. UltraESB provides weak compatibility with cloud network [40].

Mule ESB is an open source and lightweight service bus based on Java Messaging Service (JMS) and responds quickly. It easily establishes a connection and enables *apps* to exchange messaging data. However, Mule ESB does not provide service governance and security metrics. Mule ESB could be a moderate option for using a service bus for delivering services with medium performance [27].

Apache Synapse ESB is open source, lightweight, and high-performance service bus. It also provides deploying services as Apache Synapse server instances. It helps in monitoring and governance of services across the network. It also implements complex routing scenario with conditions like non-blocking HTTP/S transports for *fast-mode* of interactions. But, Apache Synapse ESB does not support composition of service components, called *module*. It offers minimum security authentication. It may result in weak federation with cloud network [49].

Red Hat JBoss Fuse is open source ESB, which is based on *Apache ServiceMix* integration platform that supports Java Business Integration (JBI) and Open Service Gateway initiative (OSGi) specification. *Red Hat Inc*. took over *FuseSource* from *Progress Software Corporation* in June 2012. Fuse ESB was renamed as *JBoss Fuse*. It is robust SOA infrastructure that delivers standardized methodology, server and tools to integrate service integration with complex applications. It supports medium performance and limited security metrics [45].

TIBCO's ActiveMatrix® Service Bus is lightweight, non-open source and high-performance ESB, which helps in organizing services by routing and transforming data formats and transport protocols. It allows fast development, nearly zero coding & low maintenance. It also manages secure communication between services and application system. It could be a good option for the commercial organization to adopt this ESB as architectural infrastructure [35].

IBM Websphere is a Java centered ESB that supports strong integration for web-based services. It is more competent, reliable, secure, and high-performance ESB. It also non-open source, permits high availability and robust failure capability to application servers. IBM facilitates with its own graphical editor, which has *drag and drop* functionality for developing mediation flows. It also helps in reducing costs with fast and flexible application service integration. This could be a better option for the commercial organization [37].

Window Azure Bus is mainly used for cloud-based services architectural platform. It is a non-open source and high-performance ESB that can also be deployed on-premise. Several services are maintained by Microsoft Corp. through Windows Azure data centers, located across the world. It provides better service governance, monitoring, and security management policy rules in handling services to facilitate its client. This could also be a good opportunity to the commercial enterprise [36].

Sonic ESB connects, mediates and controls services through its own interface environment. It is pure proprietary ESB that permits fast and secure communications with transactional failover recovery. It also provides reusability among service components in SOA system. It handles deployment service via Sonic Workbench Run Framework. It consists of different API and communications logics that are necessary to debug any errors in service integration [30].

Fiorano ESB is a JMS messaging-based ESB that manage communication interactions between multiple services. It helps in reducing complexity and increase flexibility by re-using service modules. It is a non-open source, reliable and high-performance ESB. There are various commercial products by *Fiorano Inc. (USA)* available in the market like Fiorano SOA, Fiorano Cloud, and FioranoMQ, Fiorano API Management etc., which extends different features of SOA [9].

Oracle ESB is commercial, flexible and high-performance ESB. It allows open standards to establish, transform, and route business data between multiple services. It enables governance and monitoring of business audit log data without influence on current services. It supports service orchestration with flexible integration between multiple services. It provides the secure channel for communication via routing messages in ESB with limited functionality [47]

Table 2.5 presents the information into concrete form along with additional data of control and process management of ESB product. Here, Market strategy specifies the current punch line from that particular business organization to attract its customers towards ESB product.

Table 2.5: Market overview, handling and control management for available set of ESBs

Reference	Market Strategy	Control Management	Process Handling	Comment
Oracle® **www.oracle.com**	Robust, scalable, flexible, Adapters/Transport Module	Message Tracking, Load Balancing, Message Throttling (No. of Message/time)	Support stateless message flows, secure configuration	Discuss about architectural behavior of ESB
WSO2 ESB **www.wso2.com**	Fault tolerant mediations, priority-based routing, service chaining	Governance Registry, WS-Security, LDAP, Kerberos, OpenID, SAML, XACML	Flexible logging support, Centralized configuration management	Does File Processing support
TIBCO **www.tibco.com**	Assimilate with any third party technology, Do more faster and spend less	Business Events deployment, Memory object management	Reduce complexity, enrich visibility, promote reusability	Define ESB as shared superhighway for data, messages, and events
JBoss Fuse **www.jboss.org**	Pattern based integration framework, fully certified Java™ EE platform,	OpenShift Platform-as-a-Service (PaaS), connectors for JDBC, FTP/SFTP,	Dynamic configuration and management, easily deploy or update services	Provide elastic footprint to integration beyond data centers
Apache ServiceMix **servicemix.apache.org**	flexible, open-source integration container, powerful runtime platform	OSGi-based server runtime, RESTful web services	BPM engine via *Activiti*, full JPA support via *Apache OpenJPA*,	ESB exclusively powered by OSGi
Talend ESBtalend.com/resource/esb	Powerful, Flexible Open Source ESB, Intelligent routing and mediation	drag-and-drop Eclipse environment, Talend ESB incorporates with Apache ServiceMix	Reduced costs than proprietary products	Avoids foster agility and vendor lock-in
IBM Web Sphere **www.ibm.com/wsesb**	Dynamic connectivity service solution, business agility	Service Data Objects (SDO) and Service Component Architecture (SCA)	Facilitates different protocols like JMS, SOAP, J2EE Connector Architecture (JCA) adapters	Message Logging and auditing
Fiorano ESBwww.fiorano.com	Real-time solutions addressing, obviating hardcoded dependencies	Build on a Micro-Service based architecture, agility in authentication, authorization and encryption	Message-Driven Micro-service Model, Supports sophisticated XSLT transformations	Supports dynamically configurable event-pipelines with distributed environment
AdroitLogicUltra	Development,	Support protocol like	mediation logic	provision for Zero-

ESB **www.adroitlogic.org**	Configurable and Testing for dynamic computing	XACML, JTA XA transactions, WebSockets, MLLP/S, Protocol Buffers, JSON, AMQP, SFTP	in JVM based on scripting language like JSR 223	Copy proxy
OpenESB **www.open-esb.net**	Best balance between scalability, reliability and ergonomic development	Relying on standard JBI, WSDL, BPEL, XML,SOAP, LDAP, REST, Task Scheduler	Very low TCO (Total Cost of Ownership), lightweight JBI implementation in Java.	Allows integrating with legacy systems. Initially, launched by Sun Microsystems
Petals ESB **www.petals.ow2.org**	Adaptable to large scale infrastructures, highly distributed topology and modularity	Supports JBI (JSR 208) industry specification,SOAP, BPEL, SCA, XSLT, XSD, EIP,RMI, CSV transformation	JBI pluggable components, fractal deployment framework	SOA based interconnect heterogeneous systems
Progressive Sonic ESB **www.progress.com**	Continuously Available Architecture (CAA),most Reliable and Scalable JMS-based	JAX*RS and JAX*WS annotated POJO model, support for SOAP, UDDI and WSDL	Guaranteed message delivery, dynamic routing architecture	Improved Visibility and Diagnostics, manage authentication locally

2.2.1 Criteria for comparison

This sub-section describes several criteria for the purpose of comparing existing ESBs in the market. Such categorization is based on whether a particular ESB is open source or not, provides a graphical interface to its clients, performance, flexibility and service governance. Existing ESBs could also be compared on basis of security metric, deployment scenario, virtual environment support, and essential feature like message routing, transformation, protocol conversion etc. A brief description of these criteria has been discussed below:

- *Enterprise Integration Pattern* is a standard set of integration ‘templates’ that have been used by software architects and developers in designing and implementing integration based solutions more rapidly and reliably. These patterns have been provided by every IT based company to supports its clients in service identification and accessibility [29, 41].

- *Monitoring and control* in ESB define analyzing situations for service integration and message routing during its execution. ESB controls message instance processing with mediation flow. It also evaluates overall progress of multiple services that support the deployment of applications.
- *Deployment* is a process of enabling service applications for testing a business environment. It analyzes whether a particular business logic executes its functionality to deliver better services.
- *Supports for SOA Platform* in ESB specify whether an ESB is extending the features for SOA. It also provides support for multiple services stack [33].
- *Service Governance* in ESB states how uniformly business logic rules are applied in service lifecycle management with SOA systems. Governance also defines policies through which clients can access and integrate with other services. It also incorporates security metric associated with service monitoring.
- *Service composition* defines the integration of service components, called *modules*. It provides the interaction among these modules in delivering services across the network. It also deals with supporting service discovery, configuration, modeling, and validation [8, 38].
- *Cloud adapters/connectors* in ESB are designed to provide connectivity between applications like Siebel, SAP, Oracle apps etc. It provides an interface for service components that have been utilized in the integration process. The adapter/connector never depends on ESB version, nor does it flow with ESB implementation. It just simply plug-in with ESB through well-defined connector/end-point.

Table 2.6: Comparison criteria for existing ESBs in the Market

	TIBCO Active Matrix	Fiorano ESB	Mule ESB	IBM Web Sphere	Apache Synapse	Red Hat JBoss Fuse ESB	Sonic ESB	WSO2 ESB	Talend ESB	Adroit Logic UltraESB	Oracle ESB	Window Azure Service Bus
Open-source	✘	✘	✓	✘	✓	✓	✘	✓	✘	✓	✘	✘
Graphical Interface	✓	✓	✓	✓	✓	✓	✓	✓	✓	✘	✓	✓
Visual Environment	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Supports complete SOA Platform	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Lightweight	✓	✓	✓	✓	✓	✓	✓	✓	✘	✓	✓	✓
Service Governance	✓	✓	✘	✓	✓	✓	✘	✓	✓	✘	✓	✓
Service Composition	✓	✓	✓	✓	✘	✓	✓	✓	✓	✘	✓	✓
Performance	High	High	Medium	High	High	Medium	High	High	High	Medium	High	High
Cloud Adapters / Connectors	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓
Management & Security Metric	Moderate	Limited	Limited	High	Limited	Limited	Moderate	Moderate	High	Moderate	High	Moderate
Enterprise Integration Patterns	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Monitoring & Control	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Deployment	✓	✓	✓	✓	✓	✓	✓	✓	✓	✘	✓	✓
Flexibility	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Essential features like routing, message transformation, protocol support and mediation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 2.6 depicts the association between different criteria and available set of ESB products from different companies like TIBCO Active Matrix, Mule ESB, IBM Websphere, Apache Synapse, Oracle Enterprise Service Bus, Windows Azure Service Bus and many other ESBs.

2.3 Literature Survey for Sensor Web Interfacing

Many researchers and experts have brought immiscible facts for sensor data over the web. This section extended these facts and figures associated with detailed related work in the field of sensor and web interfacing and is presented in the form of a flowchart in Figure 2.1.

In [99], Romero & Vernadat have discussed the various aspects for Enterprise Information Systems (EISs) that includes its design, networking, enterprise modeling along with its building architecture. It also presents the service integration and interoperability issues across different service instances. This work also discussed the integration patterns that support in the interoperability between different applications from different domains. It also briefs the importance of EIS in every discipline of life that grows towards the "*Internet of Everything (IoE)*".

In [100], Qiu et al. proposed IoT-enabled SHIP (Supply Hub in Industrial Park) system that characterizes the real-time interaction with multiple service components. It also developed the system with real-time tracking & tracing the object with information exchange. This wok implements the service-oriented models to handle such data. It uses XML-based data sharing cross multiple service components.

In [101], Chang et al. discussed the business process management system (BPMS) for mobile cloud computing. It also proposed a system that efficient to build the solution for utilizing resources and entities (things) across the network. It also discussed "virtual thing host" that resembles the performance of ESB for handling communication channel across the network. This system is built around IoT for delivering services.

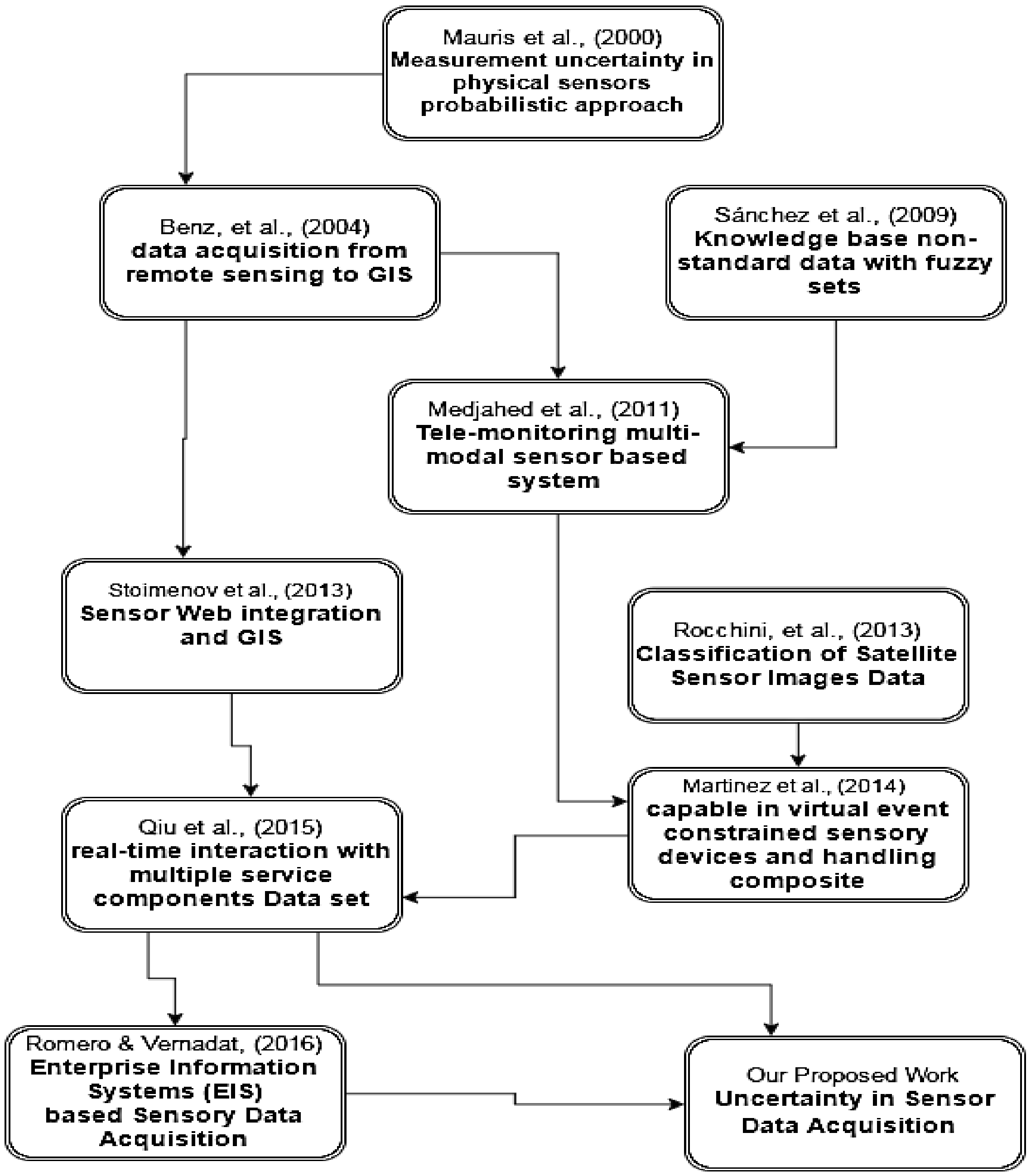

Mauris et al., (2000)
Measurement uncertainty in physical sensors probabilistic approach
Benz, et al., (2004)
data acquisition from remote sensing to GIS
Sánchez et al., (2009)
Knowledge base non-standard data with fuzzy sets
Medjahed et al., (2011)
Tele-monitoring multi-modal sensor based system
Stoimenov et al., (2013)
Sensor Web integration and GIS
Rocchini, et al., (2013)
Classification of Satellite Sensor Images Data
Martinez et al., (2014)
capable in virtual event constrained sensory devices and handling composite
Qiu et al., (2015)
real-time interaction with multiple service components Data set
Romero & Vernadat, (2016)
Enterprise Information Systems (EIS) based Sensory Data Acquisition
Our Proposed Work
Uncertainty in Sensor Data Acquisition

Figure 2.1: Systematic Flowchart for Literature Work in Sensor Data Uncertainty

In [91], Martinez et al. have modeled for virtual event sources in WSN and also represents the collection of interconnected devices through the internet which is federated to build IoT-based solution. It also discussed the Constrained Application Protocol (COAP) and Resource-Oriented

Architecture (ROA) principles for building service integration for sensor data. It also tried to prove that constrained sensor devices are capable of handling composite service as well. No data assessment is done in their proposed work.

In [103], Alena et al., (2014) briefed about the wireless architecture that constituted around Zigbee standard. In this work, it only provides the specifications for "space plug-and-play" that focused on a configuration of components in the wireless networks. It extended the proposed work with SPA-Z in which extension of SPA which offers for only wired networks. Self-configuring architectures support for fault tolerance in the WSN. It also discussed the Transducer Electronic Datasheets (TEDS) that containing sensor data and its characteristics only for its associated data format. This TEDS contain sensor data in multiple formats & style like XML, JSON, paired-key values etc, but no study is brought for its assessment and validation check.

In [95], Stoimenov et al. have discussed the ESB based solution for determining vulnerability in electric power supply network for data collection. This work presents the combination of two technologies: Sensor Web integration and GIS for predicting the fault in the transmission supply. "*GinisSense*" architecture is extended into this research work for implementing the ESB based solution. It also enables the collected data which is coupling from GIS and processed with rule-based business logic engine.

In [105], Castillejo et al. have presented the solution for guaranteed interoperability between different environments that build to form service-oriented semantic middleware. Such middleware solutions are compelled to provide integration between various service components. This implemented system collects all sensor data at developer's side which may be using devices like smartphones, computers, Pas and tablets and process at a central repository for its distributed access. This work also focused on context-aware services that create real-time IoT-based simulation software for sportsmen.

In [106], Rocchini, et al. discussed the classification of aerial photographs or satellite sensor images collected for deriving ecosystem-related mapping. This situation usually leads to uncertainty in the remote sensing tools and results in poor data acquisition. This work also presented with detail review for uncertainty occurred with ecosystems but no focused is observed for handling data with noise or error.

In [107], Medjahed et al. proposed the tele-monitoring system which was a multimodal platform for handling sensor data and provides tightly controlled datasets for elderly physiological and behavioral information. A data fusion algorithm is discussed in this work that followed fuzzy logic for controlling rules for medical recommendations. This work does not discuss the assessment of data that may incur with some noise/error.

In [109] Sánchezet al, (2009) explained the genetic fuzzy based algorithm for designing a system which provides data sets that represent information related to “linguistic granules”. This work also presented with consequence data which is used by genetic learning for dealing with fuzzy dataset. Authors also focused on knowledge-based regressive fuzzy sets for non-standard data. No data assessment and evaluation for noise/error is observed in this work.

In [112], Benz, et al. presented the strategy for analyzing the combination methods for implementing the expert knowledge system using fuzzy concept and proposed data acquisition from remote sensing to GIS. No data assessment and evaluation for noise/error is observed in this work.

In [113], Mauris et al. discussed the uncertainty in physical sensors by a probabilistic approach and proposed a fuzzy model for data acquisition in physical sensors using transformation. This approach is old and no longer helpful in collecting real-time data. This work does not discuss the assessment of data that may incur with noise/error.

2.4. Extensive Survey for Sensor and Service-Oriented System Models

This section discusses related works in the field of sensor web interfacing and how data is carried from sensor mote to desire repository (local or cloud). It is important to note that whenever data is directly stored in the repository, it includes several unwanted information which may contain noise or error. Such information need to be cleaned and filtered before storing it to the local/global repository. Figure 2.2 shows the flowchart for mentioned survey and literature review.

In [98], S. de Deugd et al. discussed the monitoring and controlling features for the environment and that implemented through interfaces for different sensor devices. This work proposed a Service-Oriented Device Architecture "***SODA***" that leverages standards from a device as well as IT assets. But, this paper does not discuss any interoperability issues related to sensor data and no observation is noticed for handling issue with noise in the sensor data.

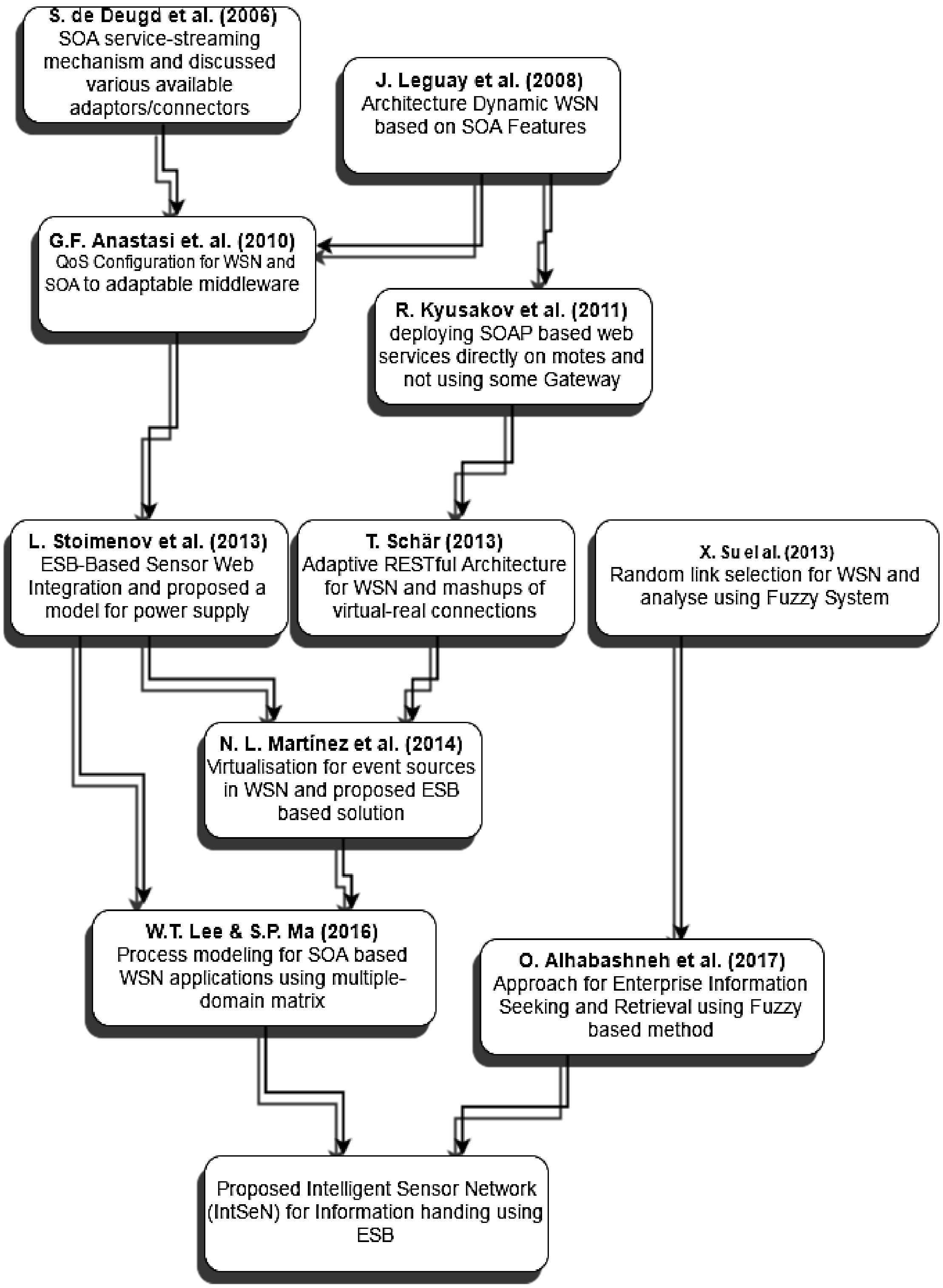

S. de Deugd et al. (2006)
SOA service-streaming mechanism and discussed various available adaptors/connectors
J. Leguay et al. (2008)
Architecture Dynamic WSN based on SOA Features
G.F. Anastasi et. al. (2010)
QoS Configuration for WSN and SOA to adaptable middleware
R. Kyusakov et al. (2011)
deploying SOAP based web services directly on motes and not using some Gateway
L. Stoimenov et al. (2013)
ESB-Based Sensor Web Integration and proposed a model for power supply
T. Schär (2013)
Adaptive RESTful Architecture for WSN and mashups of virtual-real connections
X. Su el al. (2013)
Random link selection for WSN and analyse using Fuzzy System
N. L. Martínez et al. (2014)
Virtualisation for event sources in WSN and proposed ESB based solution
W.T. Lee & S.P. Ma (2016)
Process modeling for SOA based WSN applications using multiple-domain matrix
O. Alhabashneh et al. (2017)
Approach for Enterprise Information Seeking and Retrieval using Fuzzy based method
Proposed Intelligent Sensor Network (IntSeN) for Information handing using ESB

Figure 2.2: Systematic Flowchart of Literature Work for Handling Sensor Data with Interfacing of Service-Oriented System

In [84], O. Alhabashneh et al. presented information seeking and retrieval method for systems that would able to provide context-aware information to its clients through enterprise information system. This work controls the data over the web through well-organized manner using fuzzy tolerance but fails to collect data in more realistic and accurate fashion.

In [83], Leguay et al. discussed the high-end network expedients and resource-constrained sensors by adopting SOA designs. This leads to the spontaneous triggering of tracking actions by a Linux-powered network camera and of alarms and video streams near a control room.

In [85], G.F. Anastasi et al. proposed middleware "***SensorsMW***" for sensor data monitoring using SOA. The proposed architecture in this work discussed the gathering measurement for sensor data related to physical quantity only but does not focused on an issue related to its quality assessment parameters like delay, type, and volume.

In [88], proposed a "***RESTful***" architecture which is presented with lightweight web server and JSON as a data format over Web. This work does not provide assessment and validation of sensor is carried out in this work that may lead to high relative error in the captured data. Both functionalities are presented with lightweight web server and JSON as a data format over Web.

In [91], N. L. Martínez et al. proposed a middleware architecture "nSOM" which is based on event-driven functioning for handling sensor data and monitors the interoperability for different service components. This proposed system is robust and agile for deploying capabilities for web services but no focus is made for data assessment and may lead to error in sensed data.

In [92], X. Su et al. explained the problem of dispersed fuzzy filter scheme for a course of sensor networks defined by discrete-time T-S fuzzy schemes with time-varying interruptions and several probabilistic packet losses. This work does not focus on cleaning and filtration of data before storing over the cloud.

In [94], W.T. Lee proposed a business process model and notation (BPMN) for organizing activity of Multiple-Domain Matrix (MDM) and Design Structure Matrix (DSM) based model “***BPMN-MDM***” that analyze the processed data in WSN but does not focus on assessment and relative error in captured sensor data.

2.5 Discussion

The system should be flexible and high-performance based on different features of ESB. It should allow an open standards to establish, transform, and route business data between multiple services. It must govern the agile business audit that supports service orchestration for the multiple services. It should provide the secure channel for communication via a potential message/data routing.

The above mentioned art of the review states the importance of service-oriented systems with analysis of sensor data. The sensor data must be assessed and analyzed before storing over cloud as it may incur with some deficiencies like noise/error in captured data. The discussed models present the limited conceptualization for assessment of sensor data and this motivates us to propose an “Intelligent Sensor Network (IntSeN)”. Our proposed “*IntSeN*” model presents broad view for analyzing sensor data on different parameters like delay, time, type and volume – so that more realistic and accurate data could be available for monitoring the actual environmental parameters.

Chapter3

Design and Architectural Framework for Enterprise Service Bus (ESB)

3.1. Design and Architectural Framework

ESB is a software framework that manages service and its integration with other services on a common platform. It also provides necessary infrastructure support to implement message routing, protocol translation, and message transformation. It also federates services from legacy application domain and provides the loose coupling between them. With ESB, one can design develop, deploy, and monitor services at runtime. This improves the concept of re-usability in ESB [53].

3.1.1 Essential Design Principles for ESB

Recent years have noticed a fast change in the market for ESB technology and its associated integration software. Business organizations are choosing mediation support for their architecture enhancement [4]. ESB act as a mediator that supports in interoperability, interconnectivity, and scalability of multiple services. ESB handles the message interaction through messages between multiple services. The relationship between service requesters, providers and ESB is well depicted below in Figure 3.1.

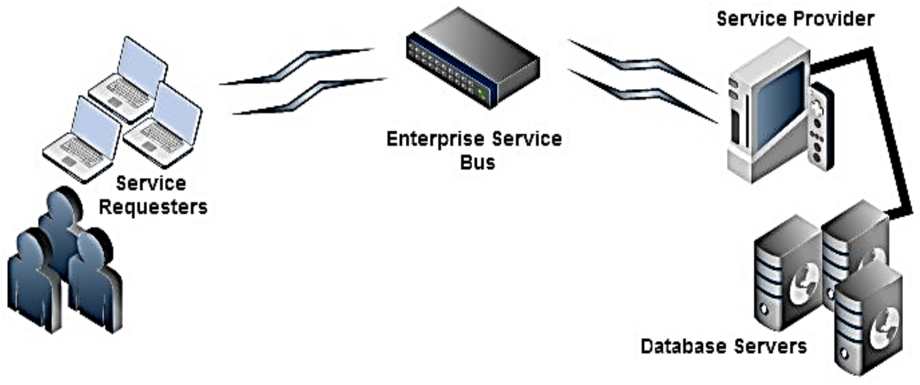


Figure 3.1: ESB Infrastructural Concept

The support for a variety of mediation allows an ESB to fulfill two core principles:

(1) Service virtualization
(2) Aspect-oriented connectivity

(1) *Service Virtualization* defines the capability of ESB to support virtually – the transmission protocol, interaction pattern, interface for communication and most importantly its identity during complete service interaction [7].

- **Transmission Protocol:** It enables ESB to process incoming request messages and transforms into the most compatible format with other service requesters. It sends and receives messages through various transport layer protocols like HTTP(S), SOAP, and JMS etc.

- **Interaction Pattern:** This allows ESB to support different patterns that help in recognizing message formats, and service integration. The ESB act as a mediator between the service provider and its associated client. This feature is also helpful in communication between two or more different ESBs together [41].

- **Environment Interface:** This permit ESB to incorporate service requesters and providers with the different interface in communication. It is also known as *"end-point"* in ESB. It also enables multiple interfaces to provide accessibility through different end-point within ESB. For example, service client and provider should not use identical interface in accessing a particular service.

- **Reconcilable Identities:** During communication, service requesters need not know its identity (like its address). It is the responsibility of ESB to manage such information (identities) for every service requester. In this way, any service requester can be served by any service provider at any time over the internet.

(2) *Aspect-oriented Connectivity* defines the connectivity issue that has always been a crucial matter in establishing communication between multiple services for SOA Systems [54]. The problem of unavailability of services may result in Distributed Denial of Service (DoS) and such issue could be avoided by adopting ESB as mediation solution. This also involves issues related to connectivity, logging and auditing of services, service validation &security, service handling & management. Such issues must be kept in mind while designing framework for any ESB [14].

3.1.2 Centric view of ESB

For Business demand in the modern era of computing, ESB could help in establishing measures for desirable solutions that also forms the business agility and robustness in delivering services across the network. These business service must be governed and monitored for better performance and it could be delivered with aid of well define service stack [56]. The service stack is set of services in service reference architecture that extends the features of simple SOA. It is well depicted in Figure 3.2.

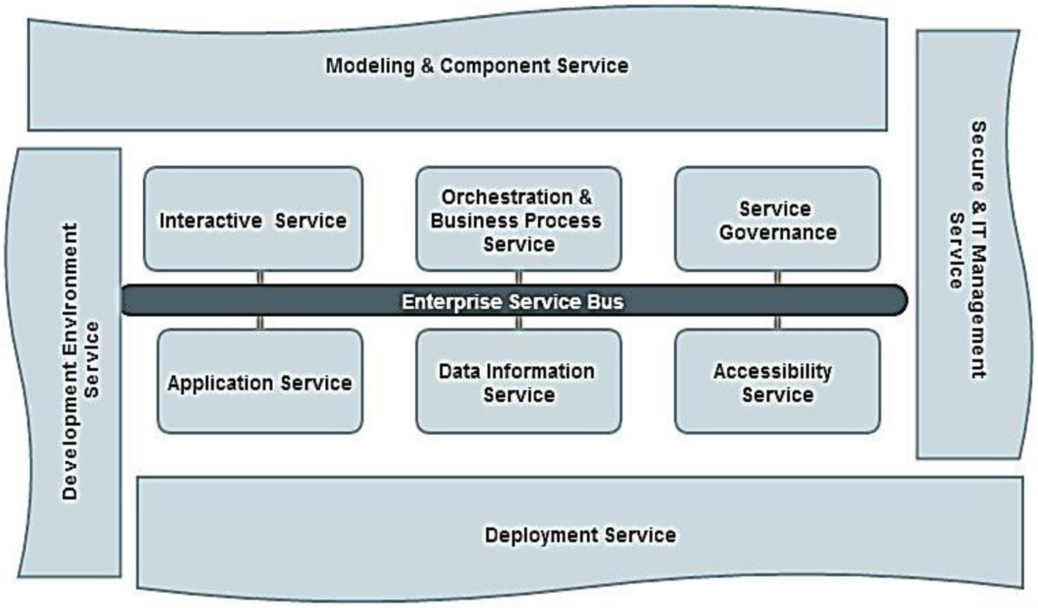


Figure 3.2: Service Stack in ESB

(a) Service Stack

ESB provides several services like– interactive services, Orchestration & Business process service, data information service, application, service, accessibility services, and service governance [13]. Such services within the ESB framework are labeled as services stack, which helps in execution and layout of different forms of services. It also provides the compatible and sophisticated development environment to service requesters for achieving on-demand services [20]. These services are as follows:

- *Interactive Services:* Such services provide essential features in delivering IT functionality associated in the interaction among services. These services support the mediation in communication for better accessibility.

- *Orchestration & Business Process Service:* Service orchestration defines as coordination between multiple services and described as a single aggregated unit to facilitate interaction among different services. These services also provide the control capabilities to manage workflow and integration.

- *Data Information Service:* Such services federate, replicate and transform information from different data sources. It also provides the necessary supports in message transformation, whenever it is invoked by some service requester.

- *Application Service:* It allows the implementation of core business functional logic that is accessible within the particular business organization. Such services are actually executed and invoked by service requesters and also support other service components in service architectural framework.

- *Accessibility Service:* It provides support in making services available to requesters. ESB act as a mediator in allowing such services that are delivered via SOA. It also helps in incorporating service with its associated data. Accessibility to the particular services governs the security policy associated with its service provider.

- *Service Governance*: It provides a monitoring to the content associated with service. It also helps to cost, planning and configuring for on-demand rental services. It guaranteed the accessibility for particular service and its scope. It provides the guideline for preparing Service Level Agreements (SLA).

(b) Secure & IT Management service

Security and management services are placed as a separate entity with ESB service stack. Since IT related services have solution wise scope and it requires better active coordination and cooperation with other service components. Security of a service is defined as preventing and protecting access to a particular service in a specific domain. Restriction on service can be limited i.e. not all requesters have the same permission to access the same service. Accessibility to particular services would be preferred on basis of policy associated with a service contract. ESB does the authorization in delivering services that can be configured to get an access. For example, web services can be spoofed and can conceal using certain fraudulent methods that used client's information. To prevent from such stealing, developers use WS-Security which is an extension to SOAP [32, 33].

(c) Development Environment service

These services help in building integration logic and connectivity issue. Equally, such services support ESB into services' discovery, registration, and publishing (in regards to service registry). With this environment, service can also be model, edit, and test and packed for deployment [16, 31]. It also allows the modification of service metadata to influence dynamic behavior of a service.

(d) Deployment Service

Deployment is the process of enabling the service application into testing or business environment. It tests the services before publishing it to constraint business environment. Further, it also binds these modules as a standard application package for deployment [13]. It also resolves the issues related to capacity, planning, testing, service hosting, service dependencies and service versioning compatibility.

(e) Modeling & Component Service

A Service component is a piece of service which helps in implementing some business logic via interfaces or endpoints in ESB. Components may interact or request other components for communication purpose; this is called as service references. It federates multiple components associated with particular service to implement loose coupling which expected to be independently deployed for particular services. These components are plug-in and plug-out with ESB [44]. It also supports the assembling of services into a group called modules. These modules actually provide complete integration solution and are well defined in Service Component Architecture (SCA). These modules work independently and do not affect the behavior of other modules [27].

Service Modeling is a concept of organizing services to fulfill the demand for service requester. It also helps in improving interaction between services. It does support the functionality like routing, transformation, and handling of messages [32]. It provides support for route discovery, validation, and transmission priorities. It logs the monitored data for interaction between services. For example, Service qualifiers define the quality of service that describes a set of communication characteristics needed by a service application.

3.1.3 What does ESB actually comprises?

ESB promises to build a decentralized architectural framework that supports several services into a unit called service bus. This integration of services defines a platform for serving requests from multiple requesters simultaneously. ESB is not an only architectural framework, but also provide highly versatile and distributed behavior [32]. Figure 1.3 illustrates actual inside conceptualization of ESB.

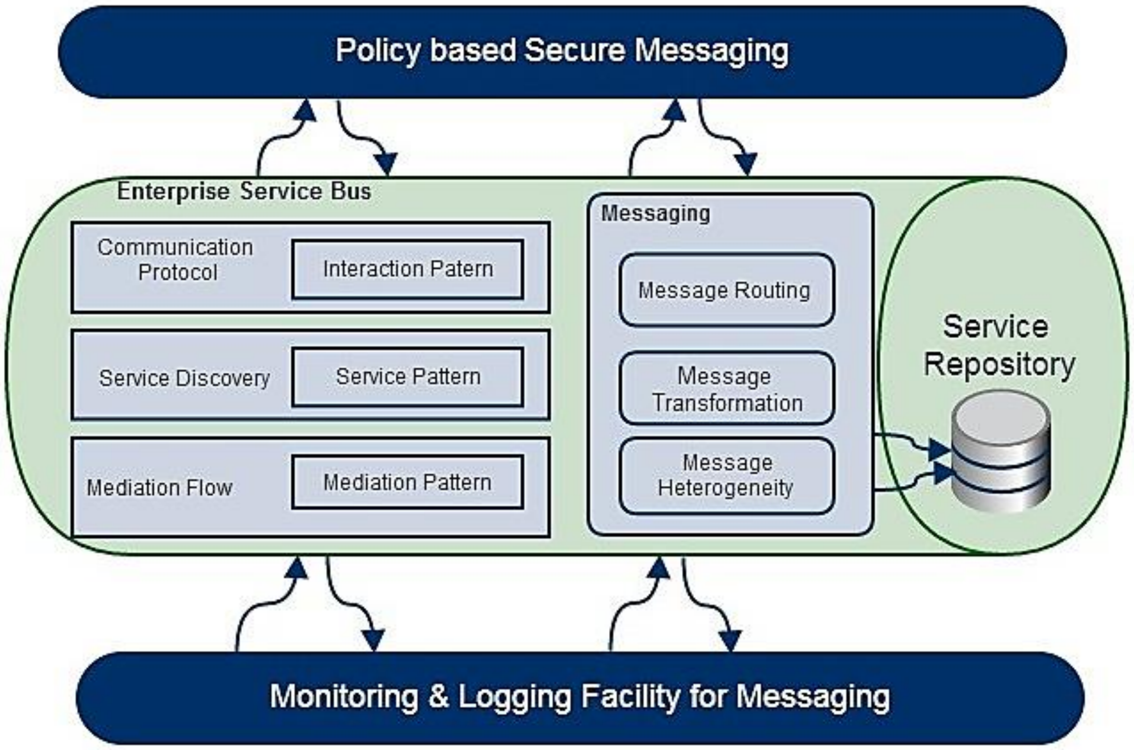


Figure 3.3: Inside View of ESB

(a) Service Repository

The service Repository (also known as *service registry*) manages the metadata that store important information related to services at the centrally located repository. It is used to encourage reusability and prevent duplication of service. The reuse of service components is a prime contributing factor to reducing cost and time as compared to other simple SOA based solutions. It also supports in task related to the scope of services and policy describing service level agreements [28]. The above Figure 3.3 depicts the position of a service registry.

(b) Policy based Secure Messaging

It supports in message interaction and communication in ESB. If a message from higher or lower layer arrives, then there must be validation checks on its access for authentication purpose. It uses digital signature and web service security-based authentication for better and secured messaging in ESB. It also involves authenticating users and access control on resources [17, 31].

(c) Communication protocol

This allows the conversion of protocols required by multiple services during communication. ESB supports various communication protocol associated with application layer like HTTP, FTP, REST, SOAP, DCOM, RFC etc. These protocols are used for establishing a connection between the service requester and provider. It also supports the necessary feature for translation of message format, which is not compatible either to the service requester or to provider [33, 34].

(d) Service Discovery

Service discovery is a part, where implemented/published service can be discovered through the service registry. This phase is used by service requester to discover implemented service. It has been observed that services face the problem of service versioning during its discovery. Service Versioning is defining as an incompatibility in recognizing service format that may not be identified by the underlying protocol [33]. The reason behind this incompatibility is unidentified message format. For example: Consider a situation in which the service provider is using version 2.0 of the particular service and that service is being requested by particular requester \`A\`, but requester \`A\` does not able to recognize the service version (non-compatible). So, there is a problem of Service versioning. In this case, ESB can be used to make transformations from an unidentified version to a known compatible version with the help of service registry.

(e) Mediation flows

The word meditation means negotiation that is conducted by some impartial party in resolving differences. To support service availability, connectivity and virtualization - mediation flow play an important role. It also helps in understanding and providing compatibility for messaging between multiple services. It monitors the workflow and processes the received messages from a requester and then forwarded to the particular service provider. Mediation varies its performance from a single interaction to multiple interactions of services in regards to reusability of services. It also uses distinct patterns of a message to its processing, also known mediation patterns. ESB uses several ready-made mediation patterns for service discovery [19].

(f) Messaging in Service Bus

ESB must support messaging through interaction between multiple services. This could help in improve availability, reliability, and flexibility related issues among services in ESB. Messaging has three important aspects related to routing, transformation, and heterogeneity as discussed below [16].

- *Message routing*: Routing provides the shortest path to deliver messages across the network. It also supports different routing transport protocols like E-mail (POP/SMTP), FTP, JMS, HTTP(S), WS-Reliable Messaging. When a message contains insufficient information, then it is difficult to route it. At this time, such message must be enriched with some additional information through service registry which maintains overall content gathering related to routing.

- *Message Transformation:* It involves the services that aggregate or enrich the XML payload of messages. It supports both XML and Non-XML data transformation that follows dynamic service selection to transform message payload based on its header content. For such transformation facilities, it utilizes XML standards such as Extensible Stylesheet Language Transformations (XSLT). It does support XML-based cache for retrieving the previously sent message [55].

- *Message Heterogeneity:* It supports multiple messaging services to maintain its workflow like synchronous & asynchronous, publish & subscribe service etc. It also provides different message formats like SOAP, JSON, XML, JMS headers, Message Format Language (MFL). Heterogeneity in the message also provides Mix and Match messaging services with WS reliable to unreliable messaging service.

(g) Monitoring & Logging facility for Messaging

Monitoring is defined as analyzing the message content during service integration at runtime. A service provider can also monitor the content for service and gather all kind of managerial associated with messaging like auditing, logging and error reporting [6]. It also prepares the statistics for messaging and its payload. It provides the real-time observation and dynamic resource loading which includes the logs and alerts that could be scanned and recorded at the organizational level.

3. 2. Message Routing Functionality in ESB

ESB primarily focuses on message routing, transformation, and its handling. It also ensures the necessary infrastructure support for reliable routing of messages between multiple services [11]. Traditional communication via ESB usually follows a rule-based mechanism for processing and configuration of messages. This technique does not provide mechanisms for dynamic routing of messages. However, there exist a lot of work in literature that extends this basic functionality through the use of Abstract Routing Path (ARP) and Abstract Routing Table (ART). A good discussion on such approaches is available in [23]. The work done in [24] is further adding the method that allowing dynamic reconfigurable routing and extends the functionality to support dynamic routing mechanism. The limitations of ESB could be overcome by allowing dynamic message routing, in which routing path is computed during at runtime.

In literature, there are two broad categories for reliable message routing:

(1) *Web-based Service Reliability:* It is an SOAP-based protocol for exchanging messages with assured delivery without its duplicity [52]. It uses an independent protocol that allows the messages to be exchanged between multiple services. Further, it provides interoperability in Web services for SOAP binding [1].

(2) *Routing Pattern:* It defines the message routing pattern in supporting service composition. However, such routing mechanism maintains static routes only and a list of alternate routes has to be provided well in advance. This mechanism also provides the solution in which method is directly applicable to the service platforms and integration tools like- TIBCO, IBM WebSphere, ServiceMix, Mule, Sonic ESB and etc. [41].

The generic functionality for ESB architecture is made up of four parts:

- *Message Mechanism:* It is the core functionality of ESB that provides handling of messages between multiple services. It also allows the services to interact with another existing service on a common shared platform of ESB.

- *Message Transformation:* It involves the conversion of a message to a particular format that can be understood by both communicating service parties. It also provides a layer of abstraction to make components as platform independent.

- *Routing Mechanism:* This process prevents the loss of messages and guaranteed its delivery. It also helps in providing a reliable route for navigating message from its clients to the appropriate service provider.

- *Message Container:* It is the actual entity which carried the data and its associated information during communication in which services are actually deployed. It also hosts the service to receive messages from other containers.

The messaging functionality associated with ESB could help in forwarding and be accepting the messages from service requesters. Such mechanism has been adopted by work in [23, 25]. Figure 3.4 pasteurizes the routing mechanism with help of ART and ARP along with business logic processing engine.

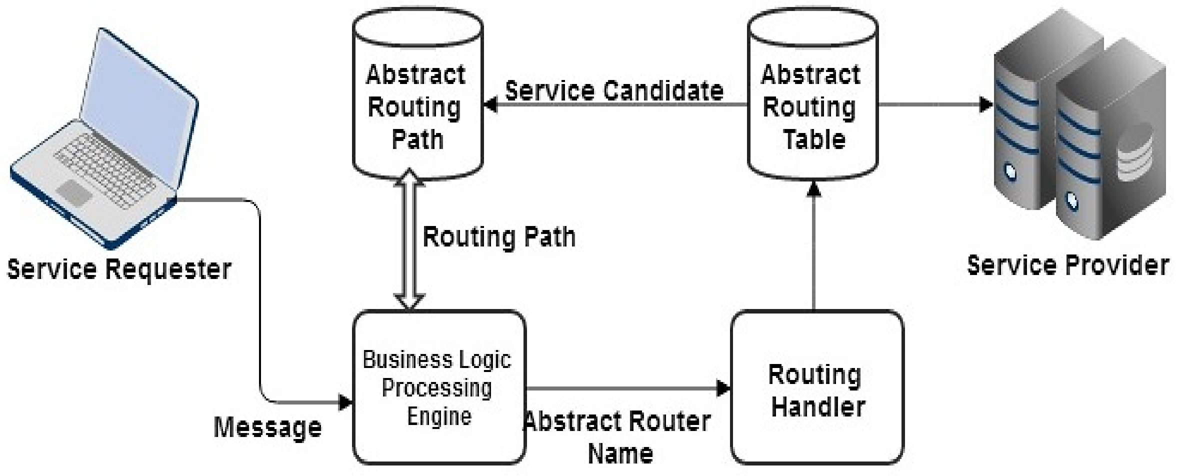


Figure 3.4: Message Routing in ESB

ARP is used to store paths for routing the messages and also provide to route maps to different abstract routers for service client and its associated provider’s URI (Uniform Resource Identifier). Generally, ARP identifies the route with help of ART in terms of *abstract service names*. It is based on the principle of ‘*store and search’* for service names and thus, determining the actual service provider. After extracting the actual address, service provider replies with this address to service request rather than just using a service name.

Business logic engine recognizes the potential routing path from ARP. Then, route handler chooses the service provider at runtime and checks the content of the routing message. Finally, route handler forwards the message to the corresponding service provider in the route map for further processing [23].

3.2.1 Classification for Message Routing

One of the major issues in ESB routing mechanism is dynamic path determination. This requires the evaluation of service that must be done at runtime to check service availability and reliability. Broadly, classification in message routing in ESB can be categorized into following three:

(1) *Content-Based Routing (CBR)*: It follows the message driven routing methodology and widely accepted as an important characteristic in message handling. In CBR, a path is identified by considering message content and that based on set of calculations for predicting route. Such criteria includes message type & its value. This category of routing mechanism involves the practice for enriching the requester’s message header with a destination address. Figure 3.5 displays CBR mechanism for communication between two - Service ‘A’ and Service ‘B’ [25].

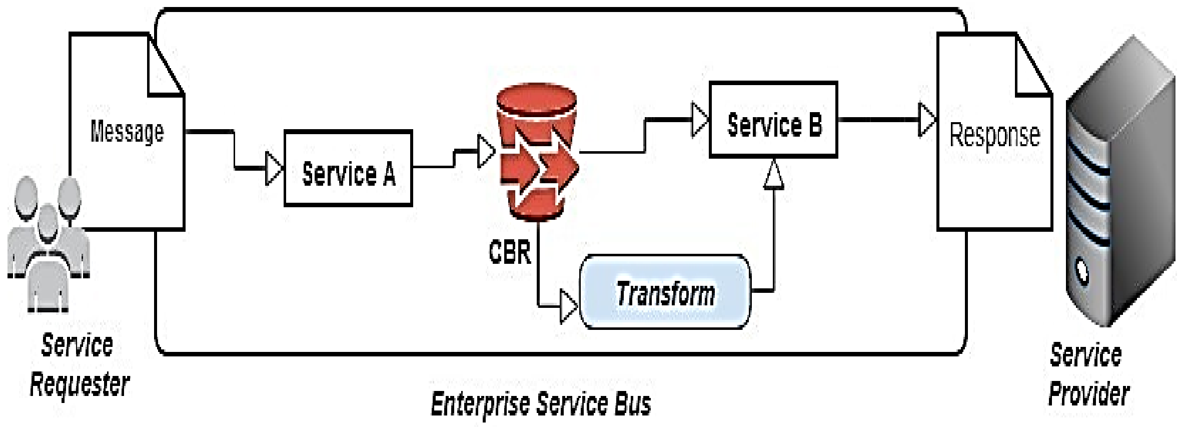


Figure 3.5: Content-Based Routing Mechanism

(2) *Pattern-Based Routing (PBR)*: Services are generally retained by containers and service composition is specified using an execution language like BPEL. This service container is also used to define the interoperability between multiple services for cooperation and coordination during an interaction. However, this methodology has a serious drawback in terms of performance, especially for complex server applications.

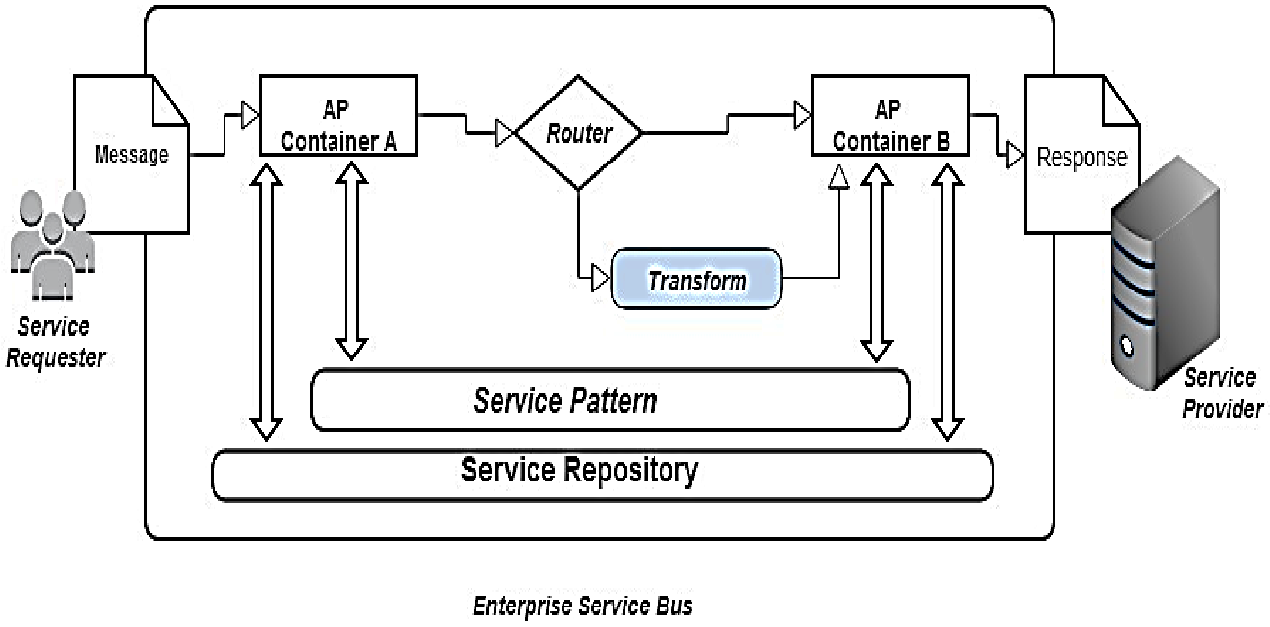


Figure 3.6: Pattern-Based Routing Mechanism

A possible solution for above problem is Pattern-Based Routing (PBR). Figure 3.6 illustrates the basic mechanism for PBR used in ESB. It introduces the *Application Patterns (AP*) that provides supporting in service orchestration. *AP* handles the messages from its service requester and then, processes it with the help of existing service patterns that have been provided from its service provider. These service patterns provide necessary support for identifying message format and its type by querying the *service pattern repository*. The *AP* container sends this processed message to *'Router'*, which transforms it with the compatible of invoked service provider.

(3) *Dynamic Routing*: This category of routing mechanism involves the path that must not be predetermined or fixed. When any client (service requester) demands a service access, then ESB assigned and allocate a unique task ID to that particular requester. Now, this task ID is assigned and inserted into the message envelope, i.e. service/message container. This container would be plug-in with ESB and *Routing Manager* navigates this message to an appropriate service provider for handling further access. But the path is determined at runtime only [23].

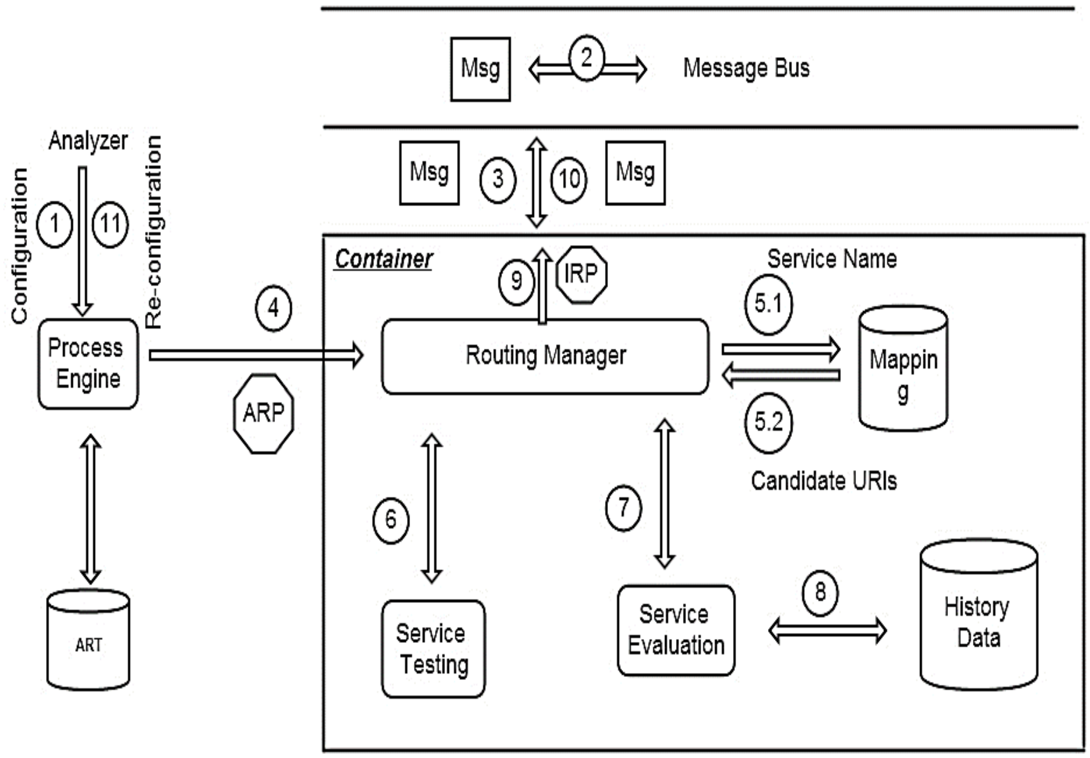


Figure 3.7: Dynamic Routing Mechanism in ESB

Figure 3.7 explains the dynamic message routing and its management used in with ESB. A brief description is given below:

1. Initially, message request goes to the analyzer component for locating the workflow. The analyzer further decides the business requirements for this request. Then, the *'process engine'* determines the number of possible paths or routes. This routing path is stored into ART.

2. During execution, messages are exchanged between multiple services and thus, ESB navigates these messages through allocated route which defined in ART.

3. Each service is plug-in through end-points which are available in ESB and engaged by the specified container to establish communication between these services. During an interaction between services, messages are, first, converted to an appropriate format and then, forwarded to further processing so that it could be easily understandable by invoked parties.

4. In next step, the container has to choose service provider at runtime. Therefore, it requests the process engine to obtain new route from the ART. After that, it extracts the service name from ART and then, creates a request for to service provider assignment.

5. Each container has its local mapping table in which it contains service names for different providers. Now, container finds the abstract service name to map its associated candidate service providers as mentioned in 5.1 and return with the list of candidate service providers' URI as shown in 5.2 step in Figure 3.7. This all would be controlled by *Routing Manager*.

6. If there are multiple services for the same functionality, then it uses specific runtime testing on each service.

7. In the evaluation step, when testing is completed, services are analyzed based on certain predefined standards like response time, incurred execution time etc.

8. The result of testing and its associated evaluation of particular service is kept into historic data.

9. Now, the container replaces the abstract service name in ARP with service provider’s URI and it creates the Instance Routing Path (IRP) for handling requested service.

10. Container, then, forwards this message to next service depending upon IRP.

11. Finally, the analyzer can change the business process by reconfiguring ARP.

3.3 Threat, Prevention & Gap with ESB

There is no standard definition for ESB but several authorities have termed ESB as middleware solution that supports service integration and its interaction with other services. To deliver, ease and fast services through ESB, software service quality and features must be rapid enough to handle the demand of business. Open-source, agility, and scalability are major challenges in setting up the software enterprise needs. To gear up with such challenges, one must be committed to open standards and interoperability from proprietary solutions. Open source software could be a better solution to avoid the vendors’ features lock-in and temporary availability as it also resolves the issue for integration and provided with more sophisticated, fast and deployable paradigm for service computing.

There is no permanent solution to the problems that occurred during service integration. While purchasing the commercial ESB solutions, it is suggested to keep the terms and conditions made during its usage [51]. At first, a decision must be taken whether particular ESB framework is sufficient and meet the entire requirement of the project at a particular organization. It is better to judge the necessity of project and which ESB best overlaps the requirement and how its complexity could affect the overall project growth.

Proprietary and open-source ESB, both are the better option depending upon the requirement associated with software building block. Sometimes, open source ESB performances are far better than commercial/ proprietary ESB [29]. At some junctions, these both ESBs have similar characteristics when compared to the computational environment of data handling like legacy systems. Thus, it is suggested at the beginning of the project, check the necessities and demand in ESB selection. There could be following threats and questions that must be kept in mind by developer or architecture designer for choosing ESB for its business:

- *Manageability:* Point out the overall handling and monitoring support with ESB. Questions like GUI support for monitoring services? How does it control the product?

- *Developer Support & Guidance:* On using particular ESB, How much technical & social support provided from its development firm? How many tutorials, articles, and videos are available? Active list or public forums available? Support would be online or offline, onsite or offsite, preferred communication language?

- *Costing:* This is a very important aspect for choosing ESB product. This may include questionnaire - What is the total cost of ownership (TCO) of particular ESB product? What does its maintenance cost? What is its subsidiary products requirement costing? What are rental charges for its cloud adapter, if required?

- *SLA Terms & Conditions:* Point out necessary condition and term for the agreement made at the time of purchase of ESB product from the particular organization. What would be guaranteed period for terms? What governing policy bears? Which part of services could be on rental basis?

- *Usability:* This point suggests the questions like - Which development environment supports? How is installation carried out? How many supporting tools are required for its installation?

- *Technical Requirement:* This point governs all the demand and requirement made and it may include the question like Offered functionality durability? Support of legacy applications?

- *Flexibility:* Brings questionnaire related to customization of ESB product. How could functionalities for a particular product be set according to own necessities?

- *Extensiveness:* Suggestion includes the possibilities for extension of existing product. This may include the questions like - Is ESB product expands to external environment? Such interface based on standards or not? Support for external API?

- *Adapter/Connectors:* Check the availability, whether developing firm of ESB provides connector/adapter for other technologies. How could customization for own adapter be possible? Would these adapters compatible with B2B products? Are these adapters fulfilling all the technologies demands?

- *Licensing:* Shows the up-gradation and transparency measures provided from developer firm of ESB Product. This may include – Which kind of license methodology is used? Is ESB product is upgradable for free or not? Is change in requirements, may change the entire license plan? Is a downgrade possible?

3.4 Discussion

This chapter concluded with features of ESB in supporting and providing facilities to services across the network. It provides the offers handling and monitoring of service that extends the features of SOA. It has been analyzed that concept of ESB is to connect various service components together i.e. s*ervice containers* that actually invoked in service execution. It is a good option for building platform in enterprise integration, as it protects and hide underlying transport protocols. It has the capability to transport messages more securely and also used to integrate several services from hybrid and homogeneous systems as a common solution for middleware. ESB also improves the architectural benefit associated with an enterprise solution for service integration and better handling of services across the network.

The Gap with ESB discusses the issues that have been found with extensive working beyond ESB. The specific business applications that are complex, dynamic and highly computational, requires a lot of effort in processing and somewhat more implementation than ESB. This introduces a new concept called - '*Integration Suite*'. This kind of Integration methodology not only offers all the features of ESB, but also includes the functionality like Business Activity Monitoring, Master Data Management, and Business Process Management (BPM). There could be additional areas that could be explored for architectural benefits in ESB, which starts with "*Microservices Architecture*". This architecture briefs about the distributed and flexibility in delivering services [48].

Today, APIs have become an actual mode of publicity to get famous of your business. The new start-up has launched its API to exposing data and internal business processes to compete for the market. Such APIs are primarily exposed using REST protocol with JSON. Nowadays, REST and JSON together resembles the combination of SOAP and XML [42]. These APIs also help in service recognition like service version, its type, and format. Usually, APIs are issued in the form of integration patterns that have been provided by its manufacturing organization. ESB could better deliver these API and help in minimizing the integration risk among services. It would also improve the Return on Investment (RoI) by adopting standards associated with ESB implementation.

Chapter 4

Mathematical Prototype for Sensor Web Integration in Service-Oriented System

Service-oriented computing is an emerging paradigm where services are platform-independent and computational entities that could be published, delivered, executed and discarded based on the quality of service (QoS) requirement [80]. It also supports the distributed, interoperable, and integration of services on a common and shared platform of ESB. This platform aims at developing a novel approach for extending the features of SOA and federates the different service components to provide the sophisticated solution for better service accessibility. This area of a service-oriented system could be merged with legacy applications like sensor network where sensed data could be accessed and delivered efficiently [78, 97]. Whereas fundamental theories and methodologies associated with data acquisition from sensor network need to address the issues that involve the interoperability and communication overhead between multiple services. This communication could be strengthened by well-defined '*link*' between correspondent '*node*'. Here, '*node*' could be a client or just sensor node which actually rises request for services access and communication between them is governed by a '*link*'. The service execution ensures the fidelity of data which is being exchanged between multiple nodes with help of desired protocol as discussed in [64].

4.1 Streaming Data for Sensor Web Integration

The streaming of sensor data could be accessed and stored locally as well as through the cloud server. Whenever, such data is stored at repository through the local server, its access is limited with a specific number of users. On the other hand, when such data is kept at a cloud, then its access is vital and rapid through various distributed algorithms and techniques. Such arrangement could be possible with different service modeling in service-oriented systems. This scenario is well depicted in Figure 4.1 in which sensor data is traffic through cloud infrastructure at global resources.

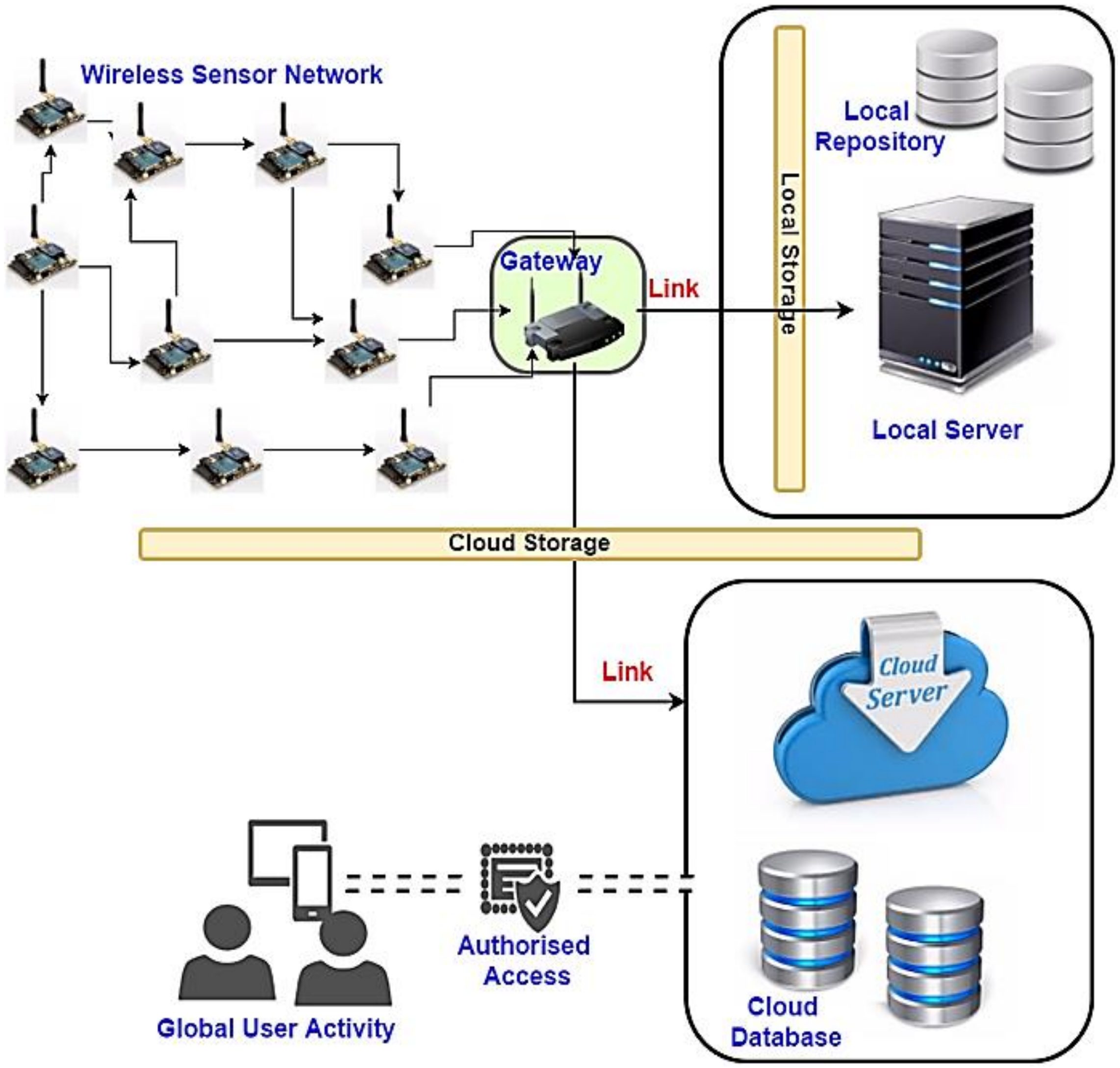


Figure 4.1: Sensor Data Streaming over Local and Cloud Server

Evolution of sensor data over the web started in 1997 at NASA laboratory, when Kevin A. Delin first coined this perception. He had also defined a wireless system as distributed, intra-connected, and communicable sensor clusters as discussed in [115]. These sensors are also installed to capture environments with variable capabilities. Kevin also pointed out the significance of these systems as “macro-instrument for synchronized sensing”. This project was successful in implementing monitoring of the environment. This idea was implemented in NASA’s Jet Propulsion Labs that could help in the battlefield surveillance purpose and later on, this concept was conceived by several applications. Thus, this integration of sensor with the web gives new hope to sensor network system which not only making sensor device intelligent but also enables the system to establish a connection with a neighboring device for exchanging data. Such system potentially shared the relevant information with cluster sensor device over data acquisition [96, 115]. This acquired information is collected and could be accessible by this web integration. The integration of two different technologies could make the system more sophisticated and efficient in delivering services across the network. This could be achieved by mean of competent middleware technology which supports the sensor data more conveniently over the web. An efficient way of handling sensor data could explore growing possibilities for our technology researcher and practitioner as discussed in [95].

For improvement in sensor and the web interfacing, Open Geospatial Consortium (OGC) was formed in 2001 that does the specification and open standards development for sensor web integration. This organization also supports rendering the geospatial services which significantly developed the standardization for a sensor device in combination with the web. OGC worked for recognition of sensor networks over the web and made an active and special working group that does standardization of sensor web interoperability issues. This working group is named as Sensor

Web Enablement (SWE). It deals with open specifications and characterization of a sensor network with web services. This SWE working group primarily dedicated for standardization for sensor data discovery, acquisition, monitoring, and event recognition. These features are well nurtured and presented in Service-Oriented Architecture (SOA) in which data is modeled using open standards and specifications. It also involves the data encodings that describe the capabilities which include data types, its mode, and phase of activation using web services.

In environmental monitoring, sensor plays an important role in capturing and sensing the different parameters. It could be possible by using different variations of sensors which scattered in different geospatial locations. This captured information is stored with help of "data logger" which is a hardware device that accountable for receiving sensed data (raw form) from the proximity sensor. This involves the interchange of data using analog to digital converter. Such inter-conversion of data is recorded into data logger on local memory until it is forwarded to a central computational unit through a well-defined gateway. The manufacturing company for such data logger offers enterprise built-in applications software (SDK) to monitor the captured data at a central computational unit. This information could be beneficial in controlling and managing deployed sensor devices at different geospatial locations. With the help of these SDKs, clients could also access and execute sensor data information at their end. It could be available via the web interface which could be customized according to client's needs. Collect raw data is analyzed and pre-processed using data cleaning techniques before making it available to client access. This information is extracted from a large set of collect and sensed data. This feature would rely on the monitoring service that assures the quality control for sensed data. Such cleaned data could be available to the data science research community group for developing standardized and open source specification in sensor web enablement [66, 94].

4.2 Architectural Service Modeling

The business conversation is an important style in the engagement of services between its clients. This could efficiently be organized through the exchange of correlated messages [63, 72]. *For example*, an insurance company client may request for a status update to his/her insurance refund. This creates an '*event*' that does the invocation and integration of multiple services. This service invocation could either commit the refund or cancel it on basis of certain predefined features. Such features are governed by service-oriented computing and it also models the service for which conversation is supported [62]. This concept can be conceived for sensor data acquisition in which the following action is required to get an update:

- *Interaction* must be based on exchanging information within the system for which typically event must occur. This may include requests, reply, commit, cancel or invoke for each interaction.
- *Interpretation* must be done using predefined business patterns, usually defined in term of enterprise integration patterns (EIP). Such procedure enables the interaction between communicating party.
- *Execution* must be governed by '*state*' of service and there has to be predefined path or route which justifies the transition of an event from one '*state*' to another.
- *Invocation* in the business conversation must be done on '*parity*' flag basis. This flag clears the interest of different '*Node'* whether it wants to interact with communicating another party.

4.2.1. Semantic Specifications

Each '*node*' is associated with other '*node*' through well-defined '*link*' that provide path or route to communicate with other entities. Data associated with sensors could be exchanged through these routes and can be streamed to cloud services. Service integration is independent of '*node*' and possible through well-defined communication *"connectors*". Such connectors provide the mapping of service component at runtime and define the features for its interactions [65, 71].

The specification of each '*link*' also define the connectors that are responsible for binding and coordinating of different service components through well-known interaction protocols as defined in sections 4.3. The service interaction associated with local and remote located cloud server could be based on two type of communication, i.e. one-way and two-way communication using connectors. The interpretation of such interaction is termed as signature interpretation in service-oriented computing in which invoked '*node'* could exchange data (sensor data) simultaneously. Hence, interaction protocols establish the generic cause for correlated sensor data exchange as discussed in [69, 73].

4.3 Formal Architectural Semantic for Sensor Data

This section presents the semantic aspect of a service-oriented computing system that does the specifications for the formal definition for sensor data to its service interaction, interpretation, and execution. It also models the execution of services that transform the status of service from one state to another with underlying interaction protocol [81]. It ensures the exchange and processing of messages (events) by invoked node in the service composition.

4.3.1. Global Resources Configuration

Global configuration consists of sensor network which is based on tuple for a simple node, associated link between them and parity checks that defines the actual status of invoked parties whether it is in ready state or not. This parity flag informs the communicating parties about the status of invoked resource. Its value varies from [0……1]. So, service configuration could be written as follow:

Cong = <Node, Link, parity>; ……..(Eq.01)

4.3.2 Service Execution configuration

This methodology discuss the exchange patterns for sensor data between invoked parties which are typically dependent on interaction. Hence, the concept of modeling associated with the conventional protocol of capturing business negotiations which mostly derives from data and its associated signature. Signature is the set of interaction names which are utilized in some business logic to determine the event associated with invoked Node.

A service execution configuration (EXE) involves the mutual disjoint set of services which are partitioned into two type of communication, either one-way or two-way. Interaction is based on this communication typology through *messaging*. It includes the different integration pattern [68].

Hence, it requires that for every n, $n' \in$ Node, if $(n, n') \notin$ Link, then

$$EXE_{(n,n')} = \phi \qquad \text{….(Eq.02)}$$

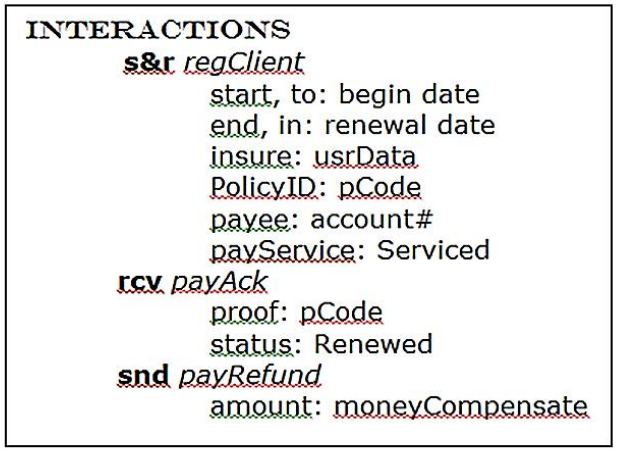
An interaction between Nodes is always in order pairs so that it could be directional and bi-directional. It could involve the transmitting of messages from one-way or two-way communication model pedagogy. Messages are forwarded through know set of Link. It is clear from the definition that if there is not such Link exists between invoked Node, there would be no messaging could possible between them.

For given two-way communication between invoked Node, set of services could be involved through messaging. Such interaction creates an event which is set of sequence instructions that must be followed by the service provider as well as by service requesters as specified in Eq.02. If requester demands any service that should be facilitated by the particular service provider, then there must be reply event from provider's side with positive or cancel reply. This means that particular service could be availed or not. Consider a case, where a reply is positive from the provider of that service and in this case, the requester could commit that service. The requester could invoke the service with desired actions. This session get expire when this event completed as discussed in [64, 71].

All such interaction must be based on parity. This interaction could be deduced with following set of instructions:

- The request-event (e_{Δ})
- The reply-event ($e_{\Re}$)
- The commit-event ($e_{\subset}$)
- The cancel-event ($e_{\not\subset}$)
- The invoke-event ($e_{\dagger}$)

Overall interaction with any service-oriented system must follow with a specified set of command between the service requester and its provider. For example, an insurance company needs to register their client with desired scheme under compensation for accidental insurance.



```
INTERACTIONS
        s&r regClient
                start, to: begin date
                end, in: renewal date
                insure: usrData
                PolicyID: pCode
                payee: account#
                payService: Serviced
        rcv payAck
                proof: pCode
                status: Renewed
        snd payRefund
                amount: moneyCompensate
```

Figure 4.2: Interaction pattern service for Insurance Company

One-way communication occurred when Node sends a single message of the request and does not get a reply from its associated service provider. This situation could also be raised with neighboring or co-located Node. The possible interaction patterns for two-way communication are depicted in Figure 4.2. *regClient*, *payAck* and *payRefund* are three services which are having associated set of instructed events.

Interactions are directional and event associated with requesters and providers: for every event $e \in$ two-way communication by requester $'r'$ and provider $'s'$, such that set $J_r(e)$ of events associated with e that can be received by $'r'$, is as follows:

- If $e \in$ two-way communication between requester $'r'$ and provider $'s'$ then,
  - $J_r(e) = \{e_{\Delta}\}$
  - $J_s(e) = \{e_{\Re}, e_{\subset}, e_{\not\subset}, e_{\dagger}\}$ ….(Eq.03)
- If $e \in$ one-way communication requester $'r'$ and provider $'s'$ then,
  - $J_r(e) = \phi$
  - $J_s(e) = \{e_{\Delta}\}$ ….(Eq.04)

4.3.3. Execution State

This subsection discusses the computational behavior with services execution configuration. This mathematical model presents the eminent view for parties which are invoked in some events. These parties are independent units and could perform computation associated with service executing in parallel and therefore, it can publish as well as execute particular events concurrently. Events are, actually, invocation of services into some constraint which needs dedicated devotion of resources and service itself. Such events are represented asynchronously by different states. Once any event is sent to some party, it is being stored into some buffer until and unless that invoked party is ready to process it [64]. Execution of event generates a sequence of states and each of which is characterized by its associated action that acquired individual party invocation in that particular interaction.

An execution state is defined X_{CONG} as tuple of $< PUB, DIV, EXC, DIS, PAR >$ ….(Eq.05)

where:

- $PUB \subseteq J$ is set of events that have been published. Its means, those events whose session is over and does not under consideration through any Link
- $DIV \subseteq J$ is set of event that are delivered at requester's site. Its means, those events which are still in a queue or under processing to be published at requester end.
- $EXC \subseteq J$ is set of an event that is executed for consideration under delivery at requester's site.
- $DIS \subseteq J$ is set of an event which is not cancel by a provider at their end and does not under any consideration so far.
- PAR is parity flag that shows the status of invoked Node

It adds parity flag in CONG as well as *executive state* for an event, so that it assures the event occurrence with the quality of service. Whenever any state change occurred, then there must be proper handling of such transition. This shows that during the state transformation of event buffering should be handed over to *Link* by *Node* with a suggestion of *Parity*.

4.3.4 Execution Route

The transition of an event from one state to another could depend on Link associated with Node [64]. There must be a path between associated Node that represents a possible execution of events which could be published, delivered, executed and discarded according to the situation as specified in (Eq.05). It is important to note that Parity is always required in transition from one state to another whose value varies from zero to one.

A route is a confined path that must be discovered by its provider $'s'$ in order to support transition of event, such that transition system $(T \rightarrow t_0)$ and $t \in T$; route $'\sigma'$ is starting in s like:

- A sequence of transition $(t \rightarrow t_1)(t_1 \rightarrow t_2).....(t_{i-1} \rightarrow t_i)$ would be finite and its final state has no further states.
- Empty path start and end with $'t'$

Above expression could also be written and formulated with $\sigma(i, i+1)$; where ith transition in sequence of events and $\sigma(i)$ is its initial state of this transition T.

Axiom 1: For $'r' \in$Node, a requester in an interaction $e \in$two-way communication, iff for each transition $t \to t^{\circ}$, when each requester can commit or cancel the deal/interaction which is offered by provider. It is important to note that requester could not do both actions simultaneously– commit and cancel the deal at same time.

Axiom 2: For $'s' \in$Node, a provider in an interaction, i.e. $e \in$two-way communication, iff for each transition $t \to t^{\circ}$, when provider ready to execute a committed/canceled deal, only one state would be allowed. The provider must reply to every request whenever request rose.

4.4 Service Interaction Semantic

This section discusses the interaction patterns between multiple services which formulate the properties based relationship either state or event through well establish Link. These properties justify the first order logic formula to specify the type of input & output for any services that facilitate the event occurrence. The efficient way to deliver such services across the computing end is ontology which sets the reason to all relations to be included in above-mentioned properties of services. Throughout this section, assume Cong = <Node, Link, parity>; and service execution configuration (EXE) with an event which is set of sequence instructions that must be followed by service provider as well as its associated requesters [70].

4.4.1 Interaction Signatures

It defines a set of interaction properties like name and its associated features which identify an event that occurred during service execution. Interaction names are actually a "*type*" that determines the properties of a particular service at an instance which derive the first order logic for K an ontology with Type T_k.

Such interaction could be fixed and variable depending on some business logic that derives the relationship between multiple services. This section discusses the fixed interaction patterns which depend on pair of <Name, Prop>; where

$< Name >$ is a type of finite and mutually disjoint set of interaction identifier which determines the interaction that based on two-way communication between requester and provider.

$< \mathrm{Pr}op >$ is set of properties which depend on event-initiation and could be expressed with $< e_{\Delta}, e_{\Re}\ e_{\subset}, e_{\not\subset}, e_{\dagger} >$

With reference to [64], interaction signatures define the importance of Node in service execution and therefore, identify the event in which Node is invoked. Such invocation of particular Node could determine the event-request (e_{Δ}) at execution time and could perform the task of publishing that event reply ($e_{\Re}$).

4.4.2 Signature Interpretation

An interpretation is a conversion of pattern involved in communication between multiple services at execution time. It depends on either communication between Node is one-way or two-way. This justifies the particular communication invoked by Node with associated signature <Name, Prop>. It can be expressed as:

An interpretation $"i"$ for signature is conversion of interaction pattern for identifier name <Name>, such that:

- for every $a \in$ $Name_{pr} \bigcup Name_{rq}$, $i(a) \in$ two-way communication
- for every $a \in$ $Name_{snd} \bigcup Name_{rcv}$, $i(a) \in$ one-way communication

……(Eq. 06)

Here, $i(a)$ is an interpretation of interaction pattern 'a' (such that $a \in$ Name) between service provider (PR) and requester (RQ) that must be followed in two communication. Whereas, interpretation of interaction pattern 'a' that occurred between multiple services within system which must be followed by one-way communication. Signature interpretation does the conversion of interaction name which is associated with service requester and provider. This behavior of conversion provides the global resource configuration with standard interaction names. So that, in future, interpretation associated with different Name could be faster and easier in aid of message exchange between multiple services.

4.4.3 Interaction Protocol

It is set of rule and regulation that must be followed in order to perform interaction between multiple services for exchanging data and information. Interaction protocol is similar to the communication occurred between two nodes from a different domain of application. Interaction signature of these two nodes plays a vital role in establishing a connection between them. It can be expressed as:

Interaction protocol $'\rho'$ is triple $\langle sig_{RQ}, sig_{PR}, ptl \rangle$ where *sig* is a signature for requester (RQ) and provider (PR). *ptl* denotes the protocol that provides the underlying rule for exchanging data and information associated to in establishing communication as discussed in [79].

Such interaction protocol must be designed in keeping the reliability of message delivery at both ends of the service provider as well as service requester. This increases the creditability of published events and reduces the abandon delay with associated Cong = <Node, Link, parity>. For example: data & information exchange, format conversion and encryption standards.

4.4.4 Communication Connector

A connector defines the interface between two Node such that {a, b $\in$ Node} and connector is triple $\langle \lambda_A, \rho, \lambda_B \rangle$ where $\rho = \langle sig_{RQ}, sig_{PR}, ptl \rangle$ is an interaction protocol, $\lambda_A : sig_A \rightarrow Node_A$ and $\lambda_B : sig_B \rightarrow Node_B$ are signature semantic mapping. This transaction would further need two interpretations i_A and i_B for $Node_A$ and $Node_B$ respectively. For interpreting the interaction protocol, it needs composition of two signatures as $(\lambda_A, i_A) \cup (\lambda_B, i_B)$ and this composition should be converse through some business rule using underlying support connector.

Let $\langle \lambda_A, \rho, \lambda_B \rangle$ be a connector for two interaction signature $Node_A$ and $Node_B$ with interpretation i_A and i_B respectively. This satisfies the connector rule which is tuple of $\langle \theta, i_A, i_B \rangle$ iff $\langle \theta, (\lambda_A, i_A) \cup (\lambda_B, i_B) \rangle$ satisfies the $'ip'$ that is interaction protocol in service-oriented computing [67].

It is important to note that the equation obtained by composition with $Node_A$ and $Node_B$ for every a $\in$ Name$_A$ and b $\in$ Name$_B$ are satisfied by $\langle \theta, i_A, i_B \rangle$ iff $i_A(\lambda_A(a)) = i_B(\lambda_B(b))$. The obtained equation justifies the connector $\lambda_A(a)$ of $Node_A$ and $\lambda_B(b)$ of $Node_B$.

4.5 Service Semantic for Sensor Data Updation

This section presents the analytical view for services and its associated specification related to the service component. It also discusses the composition of multiple services to get sensor data update using service-oriented computing as mediation. It provides the service integration and its interaction through messaging. This could be accomplished with aid of *service container*. These containers do the actual composition of multiple services on a common platform of ESB.

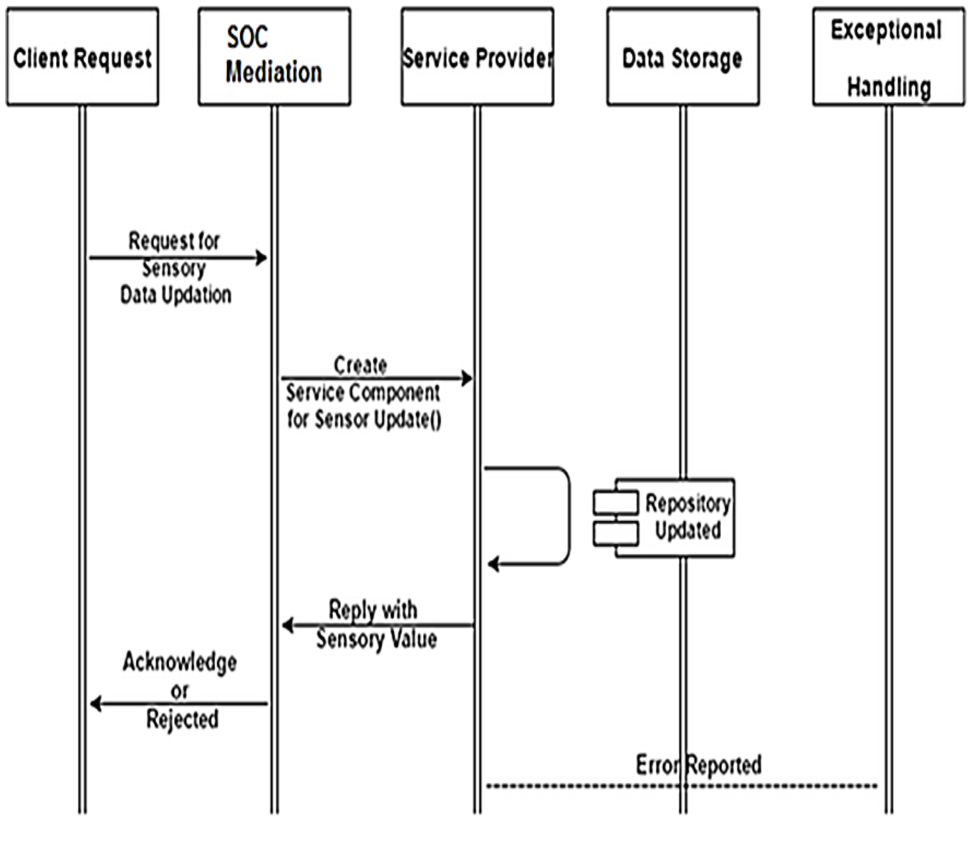


Figure 4.3: Service Flow Semantic for update Sensor Data

A mathematical algorithmic view for exchanging data in service-oriented computational architecture with arbitrary state and route could express as transition axiom and rule for service flow as shown in Figure 4.3. Let assume some execution state $'S'$ and route $'\sigma'$ for service with typical properties that for service-oriented systems, which characterize the computational feature itself.

4.5.1 Semantics for Service Container

A service container is an actual holder for the service in which a particular service could be installed and delivered to desired business logic through predefined configuration. It also provides the interface to global resource configuration that establishes a connection between given *Node*. Essentially, it uses the interaction patterns to coordinate given communication connector for a client as discussed in [70]. A service container $'\phi'$ is comprised of followings:

- Configuration conditions based on Parity bit in a pair of <Node, Link>.
- Business logic and protocol for which container has been assigned.
- Communication connector support that distinguished component responsible for the interaction. It must support $i_A(a) = i_B(b)$ iff signature semantic $sig_A \to Node_A$ and $sig_B \to Node_B$ respectively.

Table 4.1: Transition Axiom and Rule for Service Flow

$In\,general,$	$\langle state,\ route \rangle$	$\to$	$\langle S,\ \sigma \rangle$
$Assignment$	$\langle u := s,\ \sigma \rangle$	$\to$	$\langle S,\ \sigma(d) \rangle \quad for\ \ d \in D \quad [\because sensor\ \ dataset]$

$$\begin{aligned}\langle request(d),\ \sigma_R\rangle &\rightarrow \langle S,\ \sigma_R\rangle\\ &\rightarrow \langle S,\ \sigma_R(d)\rangle,\ \sigma_R(d)=\sigma'_R(D)\\ &\rightarrow \sigma(x)=\sigma'(x)\ \ for\ x\neq D\ \&\ x\neq d\end{aligned}$$

$$\begin{aligned}&Service\ Handler\ (SOC\ Mediation)\\ &(for\ each\ s\in S\ do)\\ &\quad\rightarrow create\ service\ container\ '\phi'\\ &\quad\rightarrow forward\ \langle request(d),\ \sigma_R\rangle\ to\ Service\ provider\\ &\quad\rightarrow generate\ error\ (\Theta)\ if\ not\ found\ 'd'\end{aligned}$$

$$\begin{aligned}\langle provider(d),\ \sigma_P\rangle &\rightarrow \langle S,\ \sigma_P\rangle\\ &\rightarrow \langle reply(d),\ \sigma_P := \sigma_R\rangle\\ &\rightarrow \langle cancel(d),\ \sigma_P := \sigma_R\rangle\end{aligned}$$

$$\begin{aligned}Error\ \ reported\quad \langle invoke(d),\ \sigma_P\rangle &\rightarrow \langle S,\ \sigma_P\rangle\\ &\rightarrow \langle commit(d),\ \sigma_R\rangle\\ &[\because request(d)\ stored\ \ for\ future\ reference]\end{aligned}$$

Sensor data requested by client at remote location is being first handled by service-oriented computing mediation which also acts as a *service handler*. This service handler would forward the client request to a corresponding service provider for getting the update on sensor data value $'d'$ which belongs to dataset $'D'$. This phenomenon is express in Table 4.1 which shows the transition of service from one state to another. Whenever client requests for update in sensor data, it has to be in a pair of $\langle state,\ route\rangle$.

The client request for sensor data $'d'$ in the form of $\langle request(d),\ \sigma_R\rangle$ and route $'\sigma_R'$ justifies by which client request send its data update. This request goes service-oriented computing mediation and further, it forwarded to a corresponding service provider. Then, provider analyzed this forwarded request regarding "sensor data update" and reply with a pair $\langle reply(d),\ \sigma_P \rangle$. If the service provider does not find requested data $'d'$ then it reported an error and stored it at local repository with $\langle commit(d),\ \sigma_R \rangle$.

4.6. Results & Analysis for Service Semantics

This section gives the results of experiments associated to service interaction semantic which is implemented through cloud based infrastructure. The results are compared with three different service configurations that determine the different state on service instance [66].

However, the storage of data is quite fast over local server as compare to cloud. The main reason behind such functionality is '*availability of resources*' at local level. The application is wrapped into a different set of service called *containers* and these containers interact with local server in much faster pace and response time is also rapid. Even, the connectors for such container provides better integration of services.

The access feasibility of services is analyzed in local as well as cloud server which determines the performance accuracy in regards to execution in different service states. The configuration of workstation is enabled with CPU clock speed of 3.25 GHz and 16 GB of RAM. It has been investigated in this implementation that there would be consecutive increase by 3% - 5% per clock speed in performance accuracy. The overall accuracy would reach to 97.4% when two service states simultaneously executed.

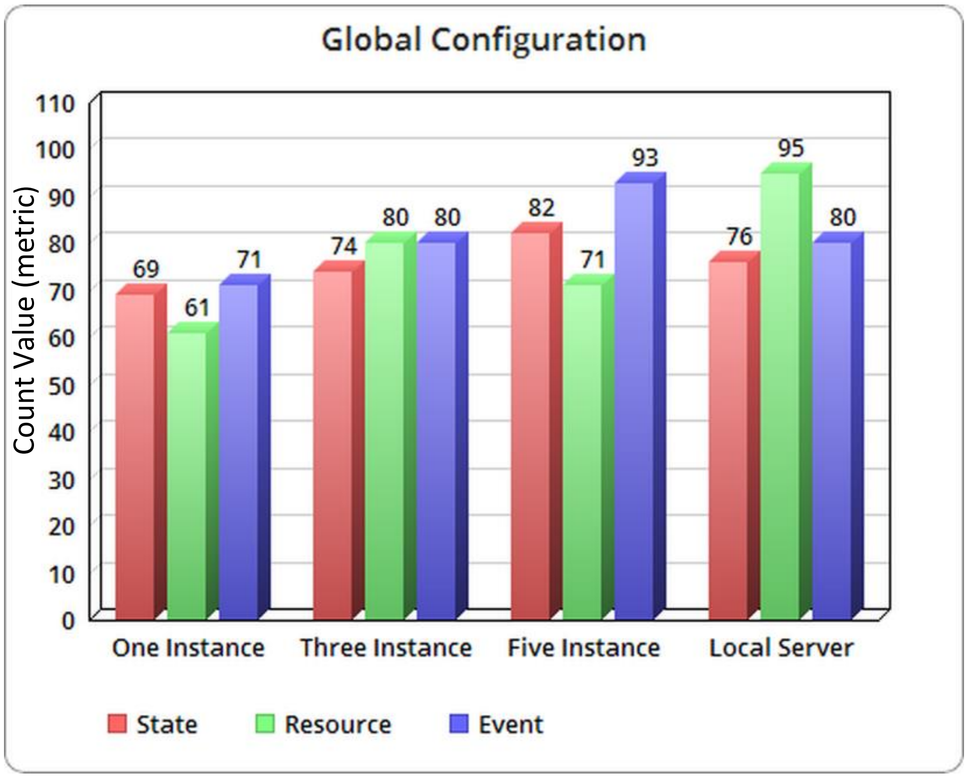


Figure 4.4: Global Service Configuration for Multiple Instances

Service accessibility at local server would be fast as compared to cloud-based servers. This methodology experiments with above-mentioned transition axiom and rules in Table 4.1. In the case of cloud-based accessibility of services, these service instances are actual entities who take the access grant for resources which are also virtually available. These service instances could be with different states and events based on their communication invocation. If any service instance is at a local server, then its associated states and events would be less as compared to cloud base server [76]. Available resources with that local server would be high as compared with the cloud server. Figure 4.4 depicts the number of states, resources and associated events in that states.

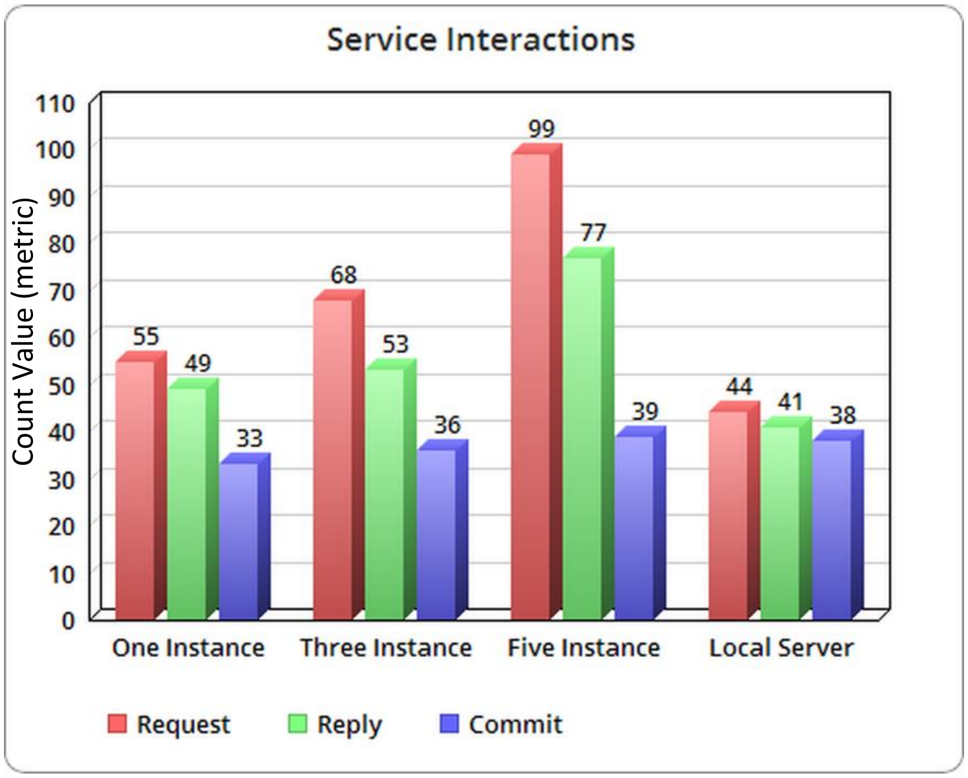


Figure 4.5: Service Interaction for Multiple Instances

The service instances interact with each other through messaging with request, reply and commit. With a local server, the number of service interaction is less to engage its corresponding resource as depicted in Figure 4.5. Consider the case of five instances of service, in which the number of sent requests are more as compared with the local server request due to its availability.

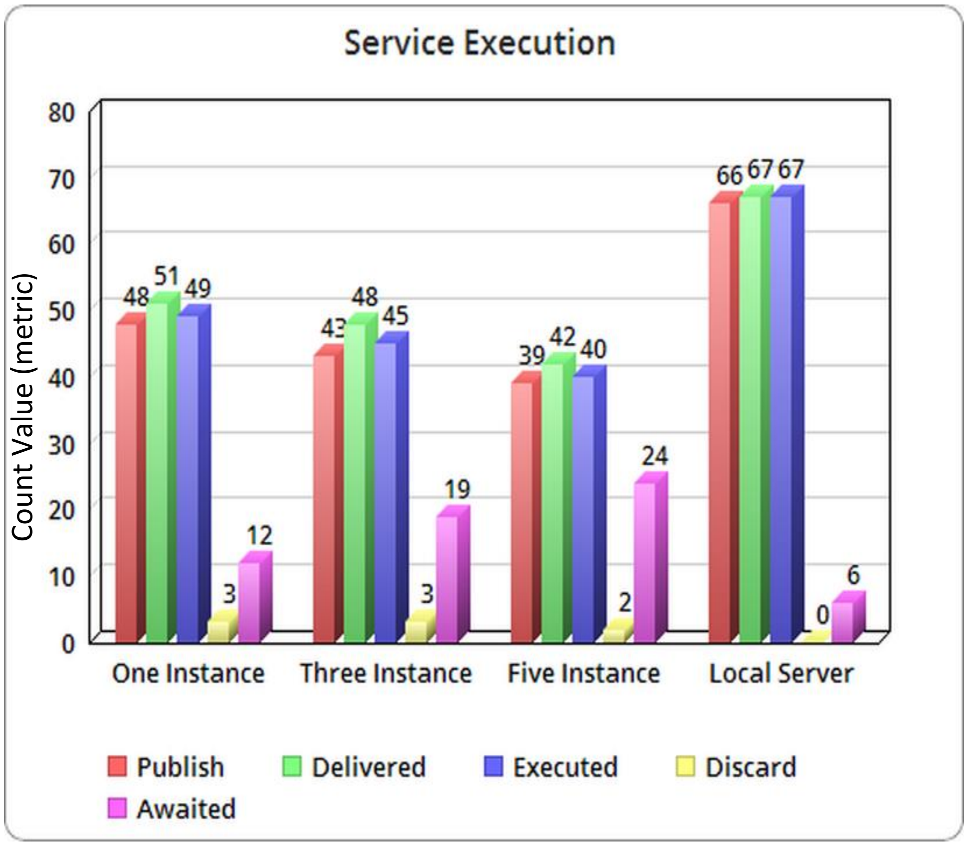


Figure 4.6: Service Execution for Multiple Instances

Any service could be in the different state depending upon its invoked event. Figure 4.6 shows the service execution in term of whether that service is published, delivered, executed, discarded or in waiting state. The number of service count in published, delivered and executed are same due to the reason of its availability as local. Whereas in cases of multiple instances, it finds different scenario depending on the number of instances.

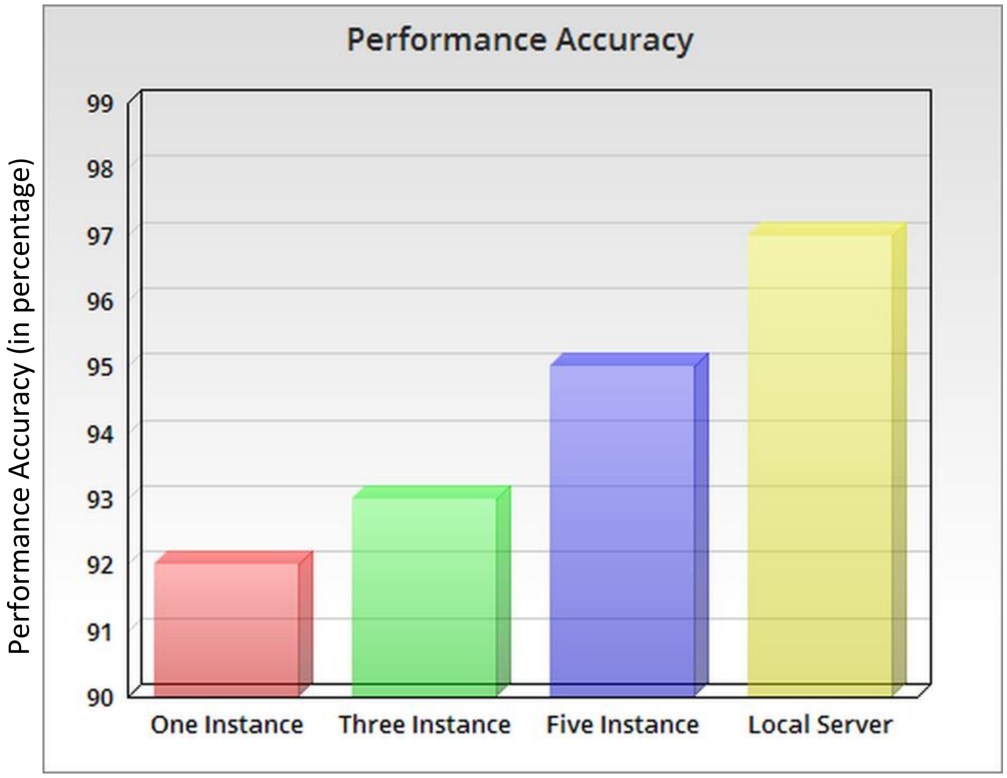


Figure 4.7: Performance Accuracy Index for Multiple Instances

The overall performance accuracy is found to be in the range of 92% to 97% approx. for the different number of instances as shown in Figure 4.7. It also depicts the traditional fact of local and global server accessibility in which resources that globally available would delay in access on the comparison of local availability of resources.

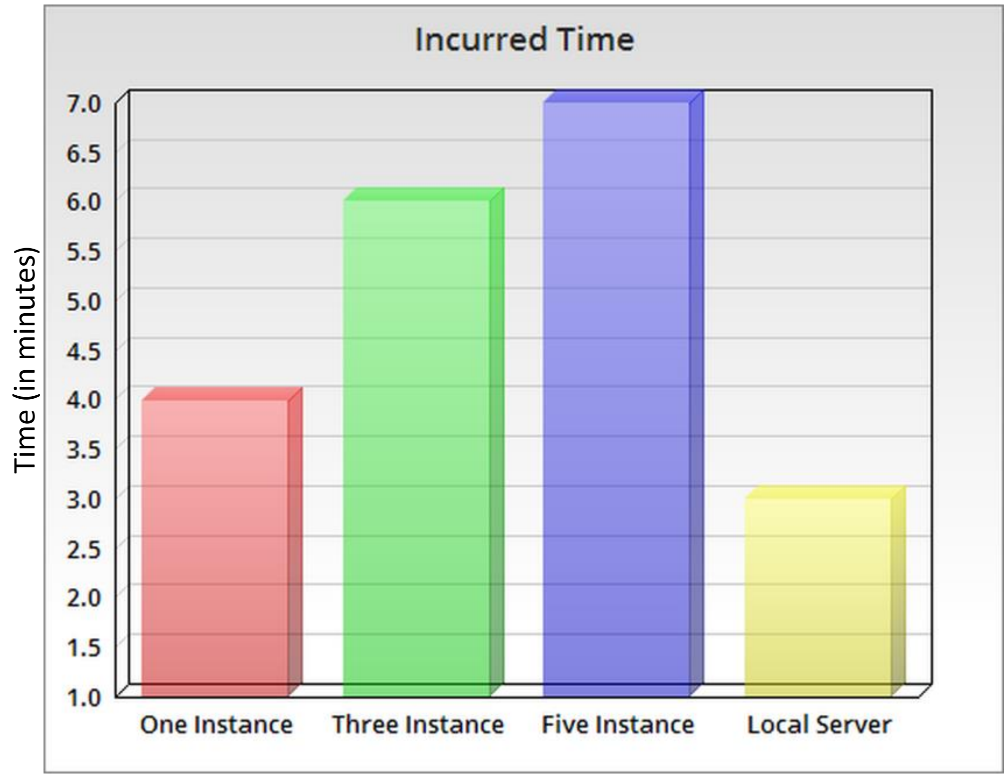


Figure 4.8: Time Incurred for Multiple Instances

Time incurred in facilitating the service instances would count in seconds at Y-axis in Figure 4.8. This result also shows the general fundamental of accessibility of resource on basis of global and local servers. It is interesting to note that services which are requested on the local server need less response time as compared to cloud servers as depicted in Figure 4.1.

4.7 Discussion

This chapter concludes the semantic perception for service execution in global configuration with sensor data acquisition. This service interaction is observed in different execution states and provides the functional and non-functional support to sensed data through the well-known gateway. The paradigmatic aspect of business interaction is explained well with the help of mathematical formulations for which services interactions are facilitated. Sensor data needs a regular update and thus, abstract semantic for such data acquisition is required to provide global accessibility of data through cloud-based services. This chapter investigates the accessibility for such dataset at two different levels; one at the local level (native server) and another at the global level (cloud servers). The discussed mathematical formulation defines possible execution state for services during one-way and two-way communications during service interaction. It also provides interaction semantics that governs the invoke '*node*' on the basis of data exchange through well-known '*link*' between them. The chapter also provides the performance based analysis for services in different states. It is important to note that services with more than three service instances, must have better performance in terms of accessibility and accuracy as compared with less number of service instances. The overall performance accuracy is found to be in the range of 92% to 96% approx. for more than five service instance. All such instances are also compared with local access utilization in different states.

Chapter 5

An Emphatic Sensor Data Handling using Enterprise Service Bus

In recent times, the era of computing has been emerged as a solution for interfacing sensors to generate data from multiple sources like mobile gadgets, actuators or devices. There is a serious demand to federate all as a sensor-based intermediate system. This intermediate based system involves the assessment and validation of data generated using sensors. It may include the processing, checks, and filtration of sensed data.

5.1 Sensor Network Interface

A sensor is a device that senses the neighboring environmental parameters like temperature, moisture, light and etc. Such device may interface to generate data that associated with monitoring and assessment of the sensor-based system. This system intermediates with service-oriented computing for a gathering of sensed data at a specified repository. To coordinate the functionality of sensors with service-oriented computation, a distributed framework is used that provides interoperability, scalability, and architectural support. ESB extends all these requirements and facilitates multiple services for integration and interaction that delivers better provision for sensor data. It also offers a sophisticated IDE that supports its designer for installing and deploying services through different service components. These service components are independent software entity that has been created to performance designated task to support underlying parent service. This could also help in achieving interaction between multiple services. Such service components actually engaged an available set of resources through virtualization. This ESB not only provides the environment for its development and deployment of multiple services but also make synchronization for its data flow during service interaction [97].

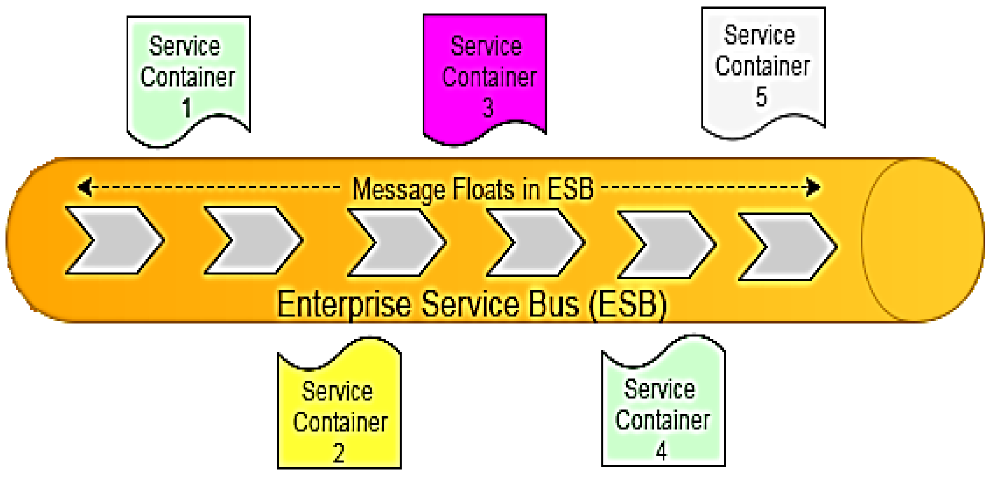


Figure 5.1: General percept for Message Handling using ESB

The data provides during communications between multiple services could be supported by a framework of ESB. It also provides synchronous of messages in a reliable and flexible structure manner. This gives rise to a loosely coupled integration of multiple services using ESB. This shows that services can be plugged-in together in a portable and vendor-neutral manner through ESB. This is well depicted in Figure 5.1.

ESB also supports pre-defined format of data patterns that must be understandable by both communicating parties. The Enterprise Integration Pattern (EIP) specifies these patterns which are provided by the manufacturer of ESB and these patterns help in recognizing services from a different domain of applications. Whenever any client request arrives at Service Engine (SE), it processes the received requests and, forwarded to its destination with appropriated route as discussed in Chapter 3. The responsibility of route dispatcher is to forward the message to desire service that is invoked in communication. Now, it is ESB which controls the flow of data information with proper synchronization. It also maintains the demand and access priority of an available set of resources that is requested by different services via its component. A service component is the entity that floats with ESB and engages the resource instance through virtualization.

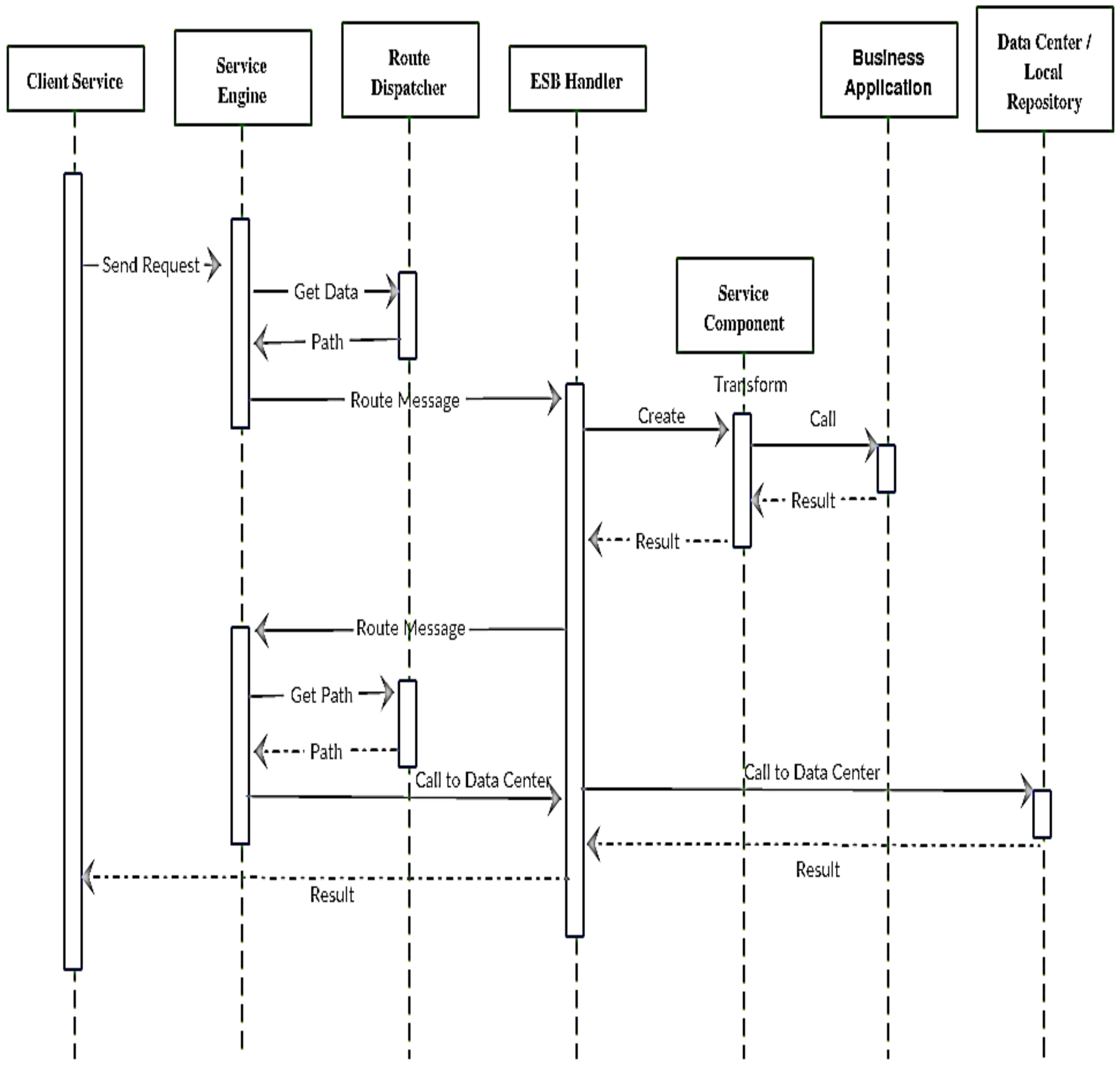


Figure 5.2: Sequence Diagram for Data Handling through ESB

The Service Engine (SE) process and execute the arrived client request to generate a path that leads the message to its destination through several steps as shown in Figure 5.2. The Service Engine (SE) through "GetData" method generates the path with help of route dispatcher and then, it routes this arrived message through generated path and passes the control to ESB. Here, ESB is capable of handling services of behalf of its service components. Moreover, these service components are itself flexible, concise and effective to implement using EIP. ESB not only integrate these components (service components) to facilitate between the service provider and requester but also provide a common and shared platform to build federated solution for improved service accessibility. It also triggers the sequence of event that calls specified function of a particular business application. The return call from specified business application provides the access calls to local data center resources [19].

5.2 Proposed Architectural Design for Intelligent Sensor Network (*IntSeN*) using ESB

The captured data is a crucial matter of concern for any sensor network that provides the collection of data at a central repository (local/global) through the well-known gateway. Such data becomes more crucial if it is carried in distributed fashion rather than collecting as a centralized mainframe manner. Here, ESB plays an important role in providing the backbone to such system architecture for handling sensor data within a particular system. It also provides the guidelines to setup strong middleware for interfacing sensor with a service-oriented system. The communication aspect of data handling would be assured by different features of ESB [96].

For traditional computing system, the vendor provides different *adapters/connectors* for new and upcoming technologies. These adapters provide a common interface like J2EE Connector Architecture (JCA) So that traditional computing could able to support new and upcoming technology. The data in such system could not be filtration due to error especially for sensor-based system in which sensed data contains noise for a particular instance of time. This chapter focuses this inability of data filtration that includes the clean and stabilizing of sensed data. The primary challenges in this endeavor are to support generic framework for accessible interface to services and to cope with heterogeneity at both endpoints of services [102].

The problem of interoperability and accessibility could be resolved by using ESB framework that includes the features like delivery assurance, underlying protocol support, message routing, and much more as discussed in Chapter 3. It also implies the improved reliability and validation of sensed data using API available in ESB. It has a wide variety of connector/adaptors which help in reducing the problem of heterogeneity at both service endpoints as an interface in ESB.

The second challenge is associated with sensed data from physical devices or sensor motes. In WSN, whenever any sensor node sends the captured data to its corresponding neighbor node or gateway (base station), it needs a proper connection and path to route this sensed data. In addition, data may contain noise or error due to voltage fluctuation or loose connection of physical devices or motes that increase the value of sensed data in an anonymous manner. This data creates an unwanted spike in sensor value and may lead to fake or ambiguous information. Such data must be cleaned and filtered before storing it at cloud repository. The major issue of data filtration and making it as stable as to store over cloud [74].

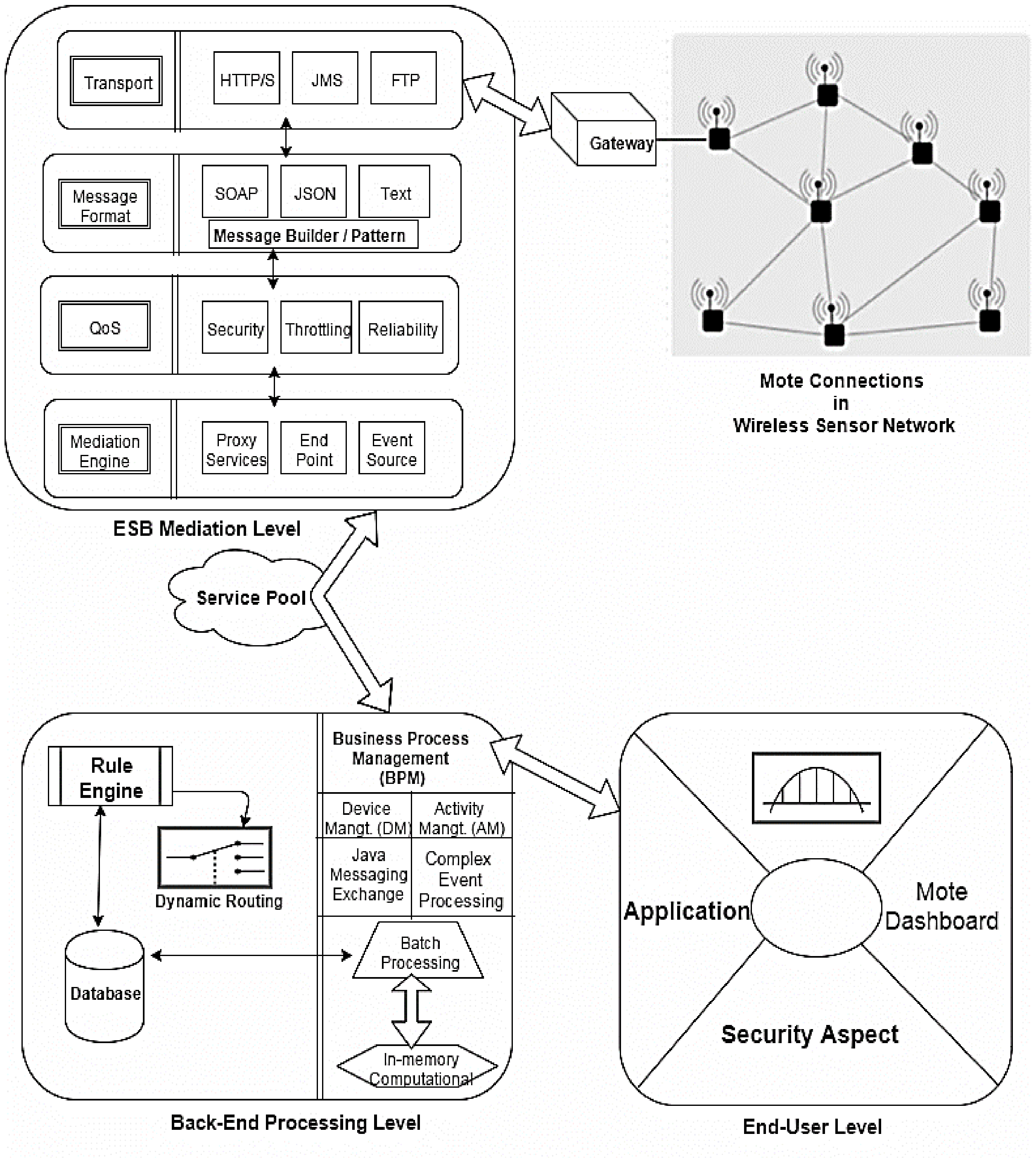


Figure 5.3: Proposed Architectural framework for Intelligent Sensor Network (*IntSeN*)

Figure 5.3 elaborates the multi-level architectural framework for sensor based system. The proposed architecture is initially divided into four levels: Wireless Sensor Network, ESB mediation, back-end processing and end-user.

This chapter proposes a multi-level based architectural framework for an intelligent sensor network (IntSeN) that is capable of handling and filtering the sensed data from multiple motes. This scenario is well pictured in Figure 5.4 as it clearly describes an overall handling of data from the sensor network and its storage at the cloud. The filtration of sensed data makes the sensor network as *intelligent* enough which data need to filter before storing to the cloud. This step saves the storage memory at cloud from unnecessary noised data which is of almost no use if it is containing noise [94].

5.2.1 Mote Connections in Wireless Sensor Network

The sensor network has gained attention in last two decades and popularly used as a solution for environmental feature capturing like temperature, humidity, light and much more. Usually, sensor motes are spread in the environment randomly and it communicates with each other through well-known routing algorithm in ad-hoc and on-demand basis. The captured data is sent from one mote to another and finally forward to its base station where it could be processed for further assessment. This data may contain some noise or error in the value and this need to filter/clean before its storage over the cloud. It could effectively be done with help of some intermediate device like *raspberry pi board* which implements some methods to clean the captured data in a moderate fashion that does not eliminate actual value data.

5.2.2 ESB Mediation Level

ESB is a middleware framework that provides mediation for service accessibility and governing architecture for an overall system. Sensor data receives in stream form which is in continuous order one after another. So, handling for such data is important and must guarantee its delivery. This assurance is provided by strengthened middleware like ESB that provide integration for multiple services over a common platform. Technically, it could be categorized into four layers:

- *Transport layer*: This layer is responsible for handling data that has been received from multiple motes through the gateway (*raspberry pi board*). It uses HTTP, SOAP, FTP and JMS protocol to control the flow of sensed data/message.

- *Message format layer*: This layer checks the different format for the message as well as data from different sensor motes in which it is received through the gateway. It also monitors whether received message format is in JSON or simple text through SOAP protocol.

- *Quality of Service (QoS) layer*: This layer adheres the quality issue with received data from sensor mote. It also validates the received data based on criteria like delay, its type, value, and volume. It does the identification whether data is containing noised or not.

- *Mediation engine layer*: This layer guarantees the connection between sensor network layer and back-end process layer where data need to be stored. It uses different adapters/connectors using proxy services and endpoints available in ESB. It also an interface between pools of service that is responsible for recognition of its format and type.

5.2.3 Back-end Processing Level

This level is responsible for storing and accessing the sensed data that have been carried from multiple sensor motes. This level is divided into two sections:

- Storage of sensed data.
- Accessibility of sensed data.

Whenever data is kept at corresponding repository or cloud, it needs to be clean and filtered before storing as it may be incurred with some unnecessary information or log. Rather than just storing all the stream data from sensor mote, it needs to be filtered based on its format type, value incurred, density and delay. This scenario can be categories as below:

- *Business Process Management (BPM)*: Whenever any request regarding data access received at BPM, it checks the authenticity of requests and then grant the permission access to sensed data. The overall interaction is governed by Java Messaging Exchange (JME) [101].

- *Batch Processing*: Sensor data is always in streaming form and need to be updated frequently as a service request for such processing requires batch computation. If all requests from clients computed into batch fashion, it requires less time and flexible adaptability in providing latest data after cleaning and filtration job [110].

- *Dynamic Routing*: This feature is one of the vibrant characteristics of using ESB as a solution for sensor based application. It routes the data packet to its desired location as it is facilitated with the routing table and route path [11].

- *Rule Engine*: when any query arises from batch computation module, then it provides the latest sensed data from existing repository. The same functionality is maintained when data is being captured/sensed by motes and it needs to be stored at a repository.

5.2.4 End-User Level

This section is designed to facilitate the client/user who actually raised the query for updated sensor data. It could be accessed through the web interface or *Android* application and it could be based on the selection of particular environmental region where sensor motes are deployed. The interface for accessing sensor data could be customized according to requirements.

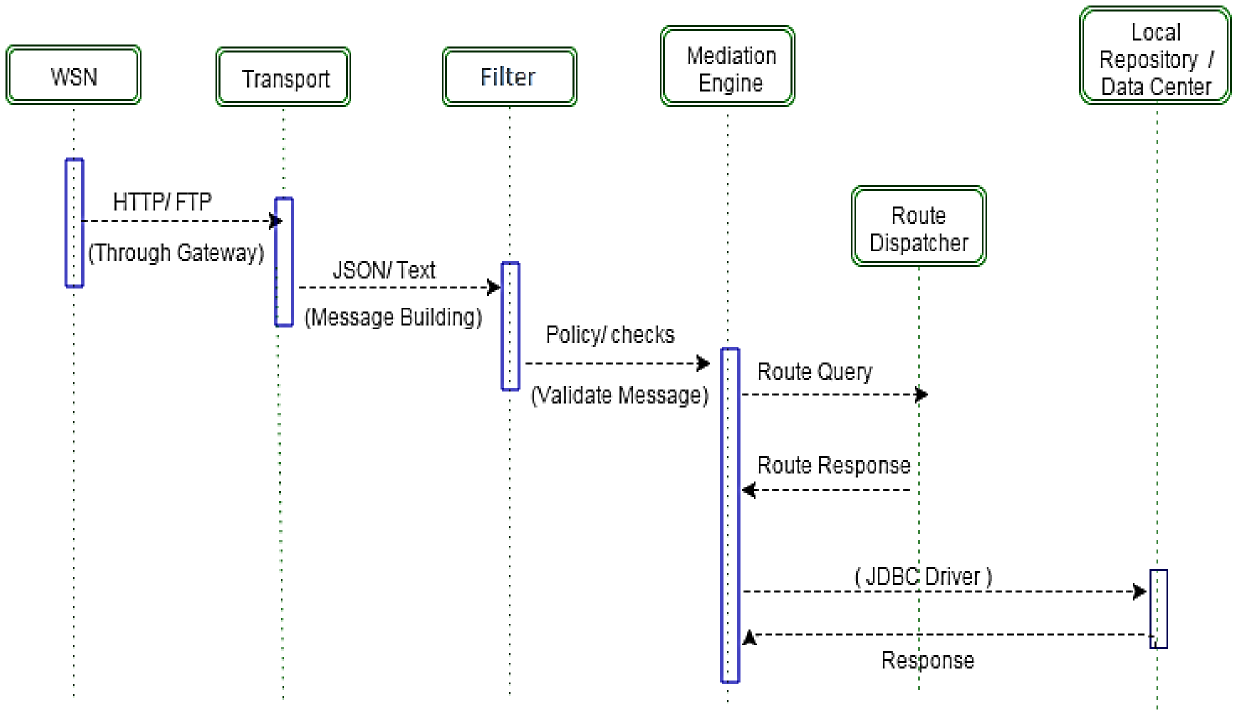


Figure 5.4 Service Workflow between Sensor Network and ESB

Figure 5.4 shows the service workflow between sensor mote and ESB. Initially, sensor mote sends the captured data through the gateway and this data is received by transport sub-layer using HTTP and FTP protocols. This data may contain noise/error and need to be filtered based on parameters like data format, delay, threshold value exceeds and density overdoes. All this process would be managed by *Mediation handler* and Raspberry Pi that provide an interfacing between it. This mediation also interacts with the repository which could be at local or global location with upcoming sensor stream data. It stores the filter data at repository through JDBC driver with J2EE connectors used in ESB.

Whenever any client raised a query for updated sensor data of particular region as discussed in Section 5.4, where such sensor motes are deployed. Such query is first received and handled by *Mediation Engine,* which sends this query to route dispatcher to locate the path for repository (local or cloud) [95]. Route dispatcher replied this query with the path to the desired repository. With the provided path, *Mediation Engine* sends the query to the repository for updated sensor data.

5.3 Result & Evaluation for Proposed Intelligent Sensor Network (*IntSeN*)

The sensed data may receive several types of unwanted data which is in the form of noise or error. Here, these data would be filtered by using mathematical techniques to get it clean and stable data. There are various types of data, which may not be in range. These data will be censored here unwanted data will be removed. The censoring is done using the following parameters:

- Density Threshold (Volume)
- Time Tolerance (Delay)
- Type Classification
- Density Threshold (Value)

5.3.1 Density Threshold (Pressure Volume)

In this section, the data are getting stable for volume parameter. When sensed data is accelerated to its base station where such data is getting over the cloud, computation logic needs to be deployed there that instantaneous increases the volume of incoming data. This random increase simply ignores useless data bits which are required to check its volume parameter. This process reduces the space consumed by the incoming sensed data and slow down the computation cycle deployed in its base station.

Methodology:

The object of this experiment is to find stabilized data on basis of pressure applied. In Figure 6, X axis is taken to represent time from 0 to 300 minutes and Y axis represents pressure amplitude 0-100 Pascal.

Red colour indicates “Raw pressure data” i.e., the pressure applied over a period of time by screw-bolt-thread machine. Green indicates stabilized output, high square green pulse width showing job is done. Green pulse is produced only when raw pressure data values exist for some times say t minutes (8 minutes is assumed here) between a particular threshold interval [t, T]. In this experiment, the minimum threshold for the pressure is taken t=21.1 and maximum threshold of T=32. So all the data below t, are assumed as 21.1 and all data above 32 are taken as 32. For the data (D), which belongs to the interval [t, T] as classified as follows.

$$D = \begin{cases} t, & D < \dfrac{t+T}{2} \\ T, & D \geq \dfrac{t+T}{2} \end{cases}$$

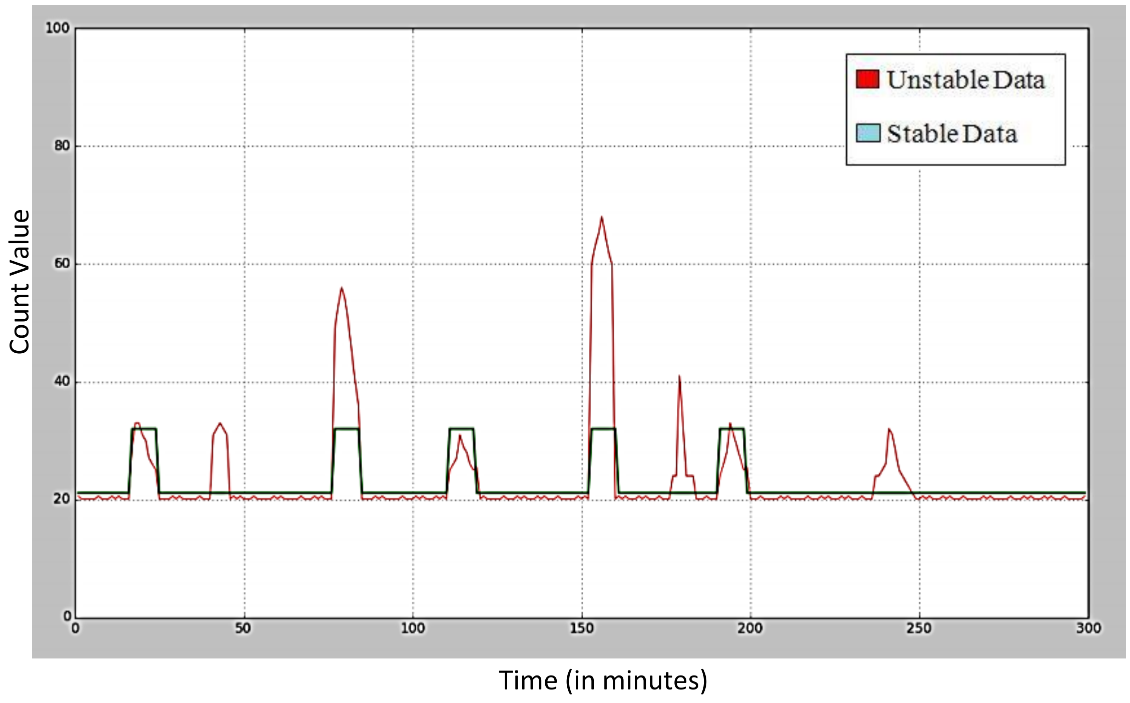


Figure 5.5: Pressure Sensor Data (Volume)

Result:

When “Red Raw data” is less than a particular volume, Volume-Parameter does not take it as a valid job. When “Red Raw data” exists for a particular volume of 08 minutes above the threshold, it is taken as a Job. The outcome of the stabilized data is shown in Table 5.1.

Table 5.1: Pressure Sensor Data (Volume)

Un-Stable pressure sensor values	**Stable pressure sensor values on basis of pressure density(pressure applied for particular time)**
20.6	21.1
20.1	21.1
20.1	21.1
20.6	21.1
20.1	21.1
33	32.0

33	32.0
31	32.0
30	32.0
27	32.0
20.1	21.1
20.1	21.1
20.6	21.1
20.1	21.1
20.1	21.1
20.1	21.1
20.6	21.1
20.1	21.1
20.1	21.1
20.1	21.1
49	32.0
53	32.0
56	32.0
54	32.0
50	32.0
45	32.0
40	32.0
36	32.0
20.1	21.1
20.1	21.1
20.1	21.1
20.1	21.1

5.3.2 Time Tolerance (Delay)

When sensed data is forwarded to some base station through a well-known gateway, it may suffer through data link breakage and inline power fluctuations. Due to this problems, raw sensor data contains sudden amplitude spikes and discontinuity in data packets. This data need to be cleaned and stabilized using tolerance limit in fuzzy theory.

Methodology:

The object of this study is to stabilize the temperature sensor value depending upon last and first received a value which is immediately before and after the data lost. This could be achieved by adopting specified tolerance value. For, time tolerance, one range (interval) of timings is fixed and all the data are accepted within the time range. All the data outside the range are invalid. But, every real field, some tolerances could be allowed. Let $I = \{I_1, I_2, ..., I_n\}$ be a finite family of fuzzy intervals on the real line which represents the time intervals and $\tau = \{\tau_1, \tau_2, ..., \tau_n\}$ be the corresponding fuzzy time tolerances. Now, the membership values ($\mu_{i,j}$) of intersections with time tolerance could be found from the following formula.

$$\mu_{i,j} = \begin{cases} 1 & , if\ c(\tau_i \cap \tau_j) \geq \min\{c(\tau_i), c(\tau_j)\} \\ \dfrac{s(\tau_i \cap \tau_j) - \min\{s(\tau_i), s(\tau_j)\}}{s(\tau_i \cap \tau_j)} h(\tau_i \cap \tau_j) & , otherwise\ if\ s(\tau_i \cap \tau_j) \geq \min\{s(\tau_i), s(\tau_j)\} \\ 0 & , otherwise \end{cases}$$

Where $c(I_i \cap I_j)$ is the core of the intersection of the intervals I_i and I_j. $\min\{c(\tau_i), c(\tau_j)\}$ is the minimum of the cores of the corresponding tolerances τ_i and τ_j. Also, $s(I_i \cap I_j)$ and $\min\{s(\tau_i), s(\tau_j)\}$ are the support of the intersection of intervals I_i and I_j along with minimum support of tolerances τ_i and τ_j respectively. $h(\tau_i \cap \tau_j)$ is height of the membership value in time tolerance τ_i and τ_j respectively.

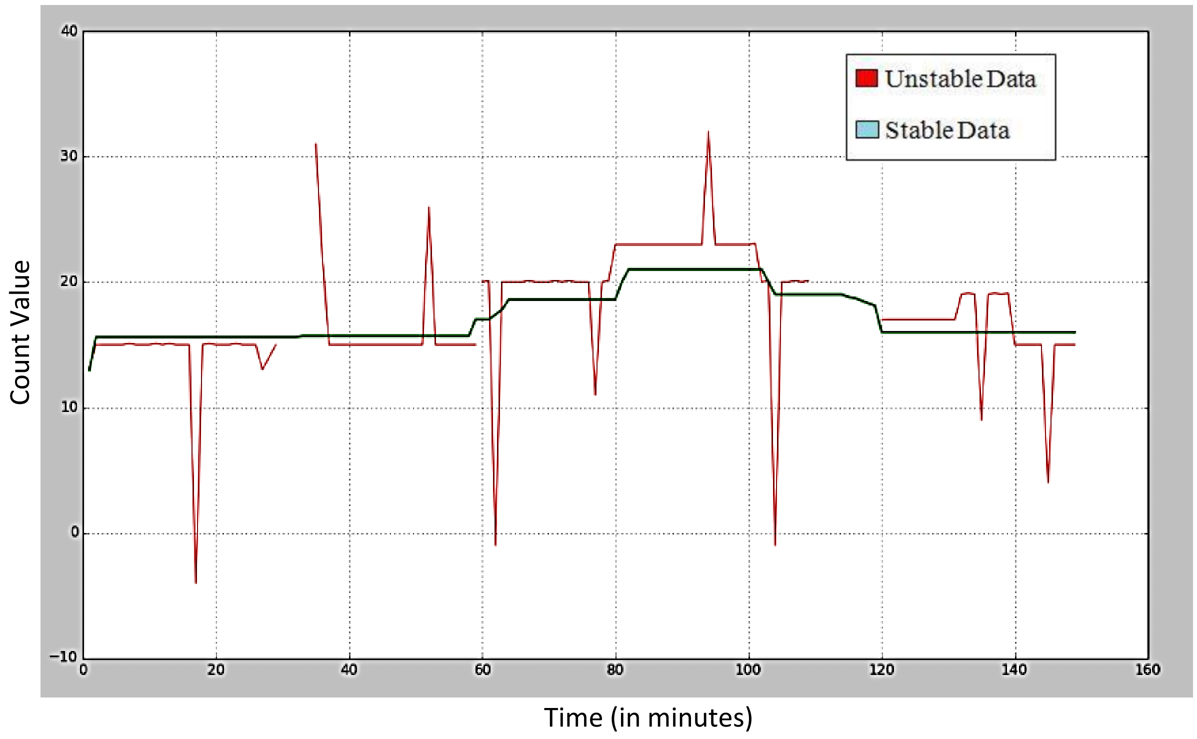


Figure 5.6 Temperature Delay Sensor Data

The object of this study is to produce average temperature value while loss in data packets depending upon last received and first received packet before and after the lost packet. Here, in Figure 5.6, X axis is time from 0 to 160 minutes and Y axis is temperature value in degrees 0-40 degrees. Red indicates raw temperature data, collected over span of 160 minutes. Temperature data packets are randomly lost due to power loss at the sensor unit. Raw data is full of glitches and lost packets. Green indicates stabilized output, average temperature values are calculated depending upon last and first received data values before and after data loss. If raw temperature values are too big in amplitude due to random glitch in line, they are ignored while computing average.

<u>Result:</u>

Final temperature values gradually increase and decrease despite of suffering packet loss, lost data packets are recovered by calculating average of last received and first received temp values prior and post to lost data packets temperature values. The outcome of the stabilized data is shown in the Table 5.2.

Table 5.2: Temperature Sensor Data (Delay)

Un-Stable temperature data on basis of Data Type having NON-Numeric data	Stable temperature data on basis of Data Type ignoring NON-Numeric data
15.0	15.6
15.0	15.6
-4.0	15.6
15.0	15.6
14.0	15.6
15.0	15.6
31.0	15.7
22.0	15.7
15.0	15.7
26.0	15.7
15.0	15.7
20.0	17.0
20.1	17.0
20.0	17.8
20.0	18.6
11.0	18.6
20.0	18.6
20.1	18.6
23.0	18.6
23.0	21.0
32.0	21.0
23.0	21.0
23.1	21.0
20.0	21.0
20.1	20.0
-1.0	19.0
20.0	19.0
20.0	19.0
20.1	19.0
17.0	16.0
17.0	16.0
17.0	16.0
19.0	16.0

19.1	16.0
19.0	16.0
9.0	16.0
19.0	16.0
19.1	16.0
19.0	16.0
15.0	16.0
15.0	16.0
4.0	16.0
15.0	16.0
15.0	16.0

5.3.3 Type Classification

The average increase and decrease in temperature values only happens when raw data has changed for a particular time span. Problematic situation is created when sudden rise in temperature is observed over a particular period. Immediate spikes are ignored due to power fluctuation and inline signal attenuations at sensor mote. This may affects the type for data format in which it is bound to receive ideally. It arises due to attenuations in the sensed data and it changes the polarity for data type. Such attenuation need to be stabilized by tapering it to threshold value.

Methodology:
In Figure 5.7, X axis is represented as a time from 0 to 275 minutes and Y axis is represented as temperature value from 0-100 degrees. Here red indicates raw temperature data, consisting of numeric and non-numeric data. Temperature data is accompanied by Non-numeric ASCII characters and DC voltage glitches. Green indicates stabilized output, average temperature values are calculated by ignoring non-numeric data. Sudden sign changes, random voltage glitches, and non-numeric data is constantly ignored. The result may be concluded as final temperature values gradually increase and decrease despite suffering non-numeric voice and voltage glitches in line.

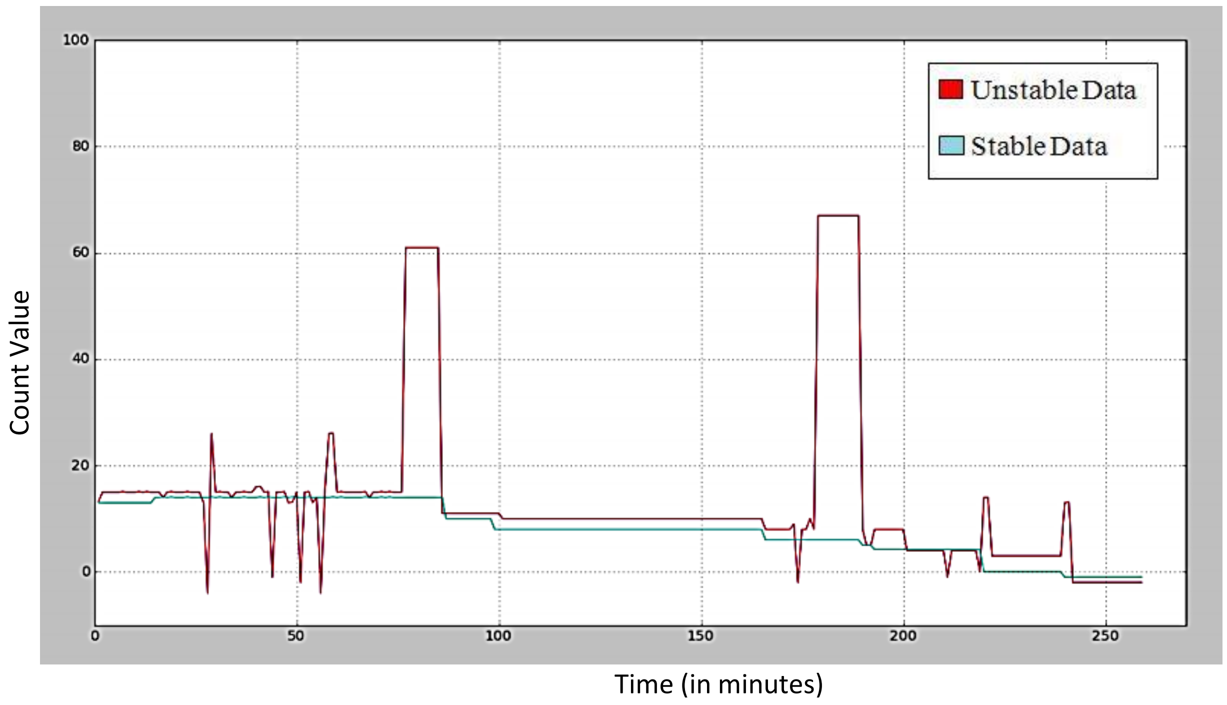


Figure 5.7 Temperature Sensor Data

<u>Result</u>:

The data which is not well-known could be treated as invalid and dropped. Object of this study is to stabilize the average temperature value while ignoring non-Numeric values. Here, raw data may be non-numeric and negative numeric value that could completely be ignore. If data is with threshold numeric value and suffers from sudden fall or hike, then it must be ignored. Even, if data is numeric but with different sign after gradual fall, would also be treated as valid. This scenario is well depicted in Table 5.3.

Table 5.3: Temperature Sensor Data (Type)

Unstable temperature data suffering from sudden data packet loss	**Stable temperature data by computing average increase and decrease in values during data packet loss**
13.2	13.0
15.0	13.0
15.0	13.0

15.0	14.0
13.0	14.0
-4.0	14.0
15.0	14.0
26.0	14.0
26.1	14.1
14.0	14.0
15.0	14.0
15.1	14.1
61.0	14.0
61.0	14.0
11.0	14.0
11.0	10.0
10.0	8.0
10.0	8.0
8.0	6.0
8.0	6.0
9.0	6.0
-2.0	6.0
8.0	6.0
8.0	6.0
10.0	6.0
8.0	6.0
67.0	6.0
67.0	6.0
8.0	5.0
5.0	5.0
5.0	5.0
4.0	4.2
4.0	4.2

5.3.4 Density Threshold (Value)

In general, threshold means some certain amount. A threshold is called a fuzzy threshold if there exists non-negative real number T such that $\sum_{u \in U} \sigma(u) \leq T$ if and only if $U \subset V$ is stable set. Here, signa values are the membership values of data and T is set as a threshold values.

Basically, a sudden spike in temperature value appeared due to link breakage and inline power fluctuations. This problem adds attenuations in sensor data which could add discontinue in data packets. Such data need to be cleaned and stabilized using tolerance limit in fuzzy theory.

<u>Methodology:</u>

The object of this study is to produce average temperature value while ignoring sudden rise and fall in temperature values. X axis represents time from 0 to 200 minutes and Y axis is represented as temperature value in degrees 0 to 70 degrees. Red indicates raw temperature data, collected over the span of 200 minutes as shown in Figure 5.8. Raw value consists of Temperature data values that suddenly rise due to voltage glitches. Green represents stabilized output, average temperature values are calculated by ignoring sudden data peeks.

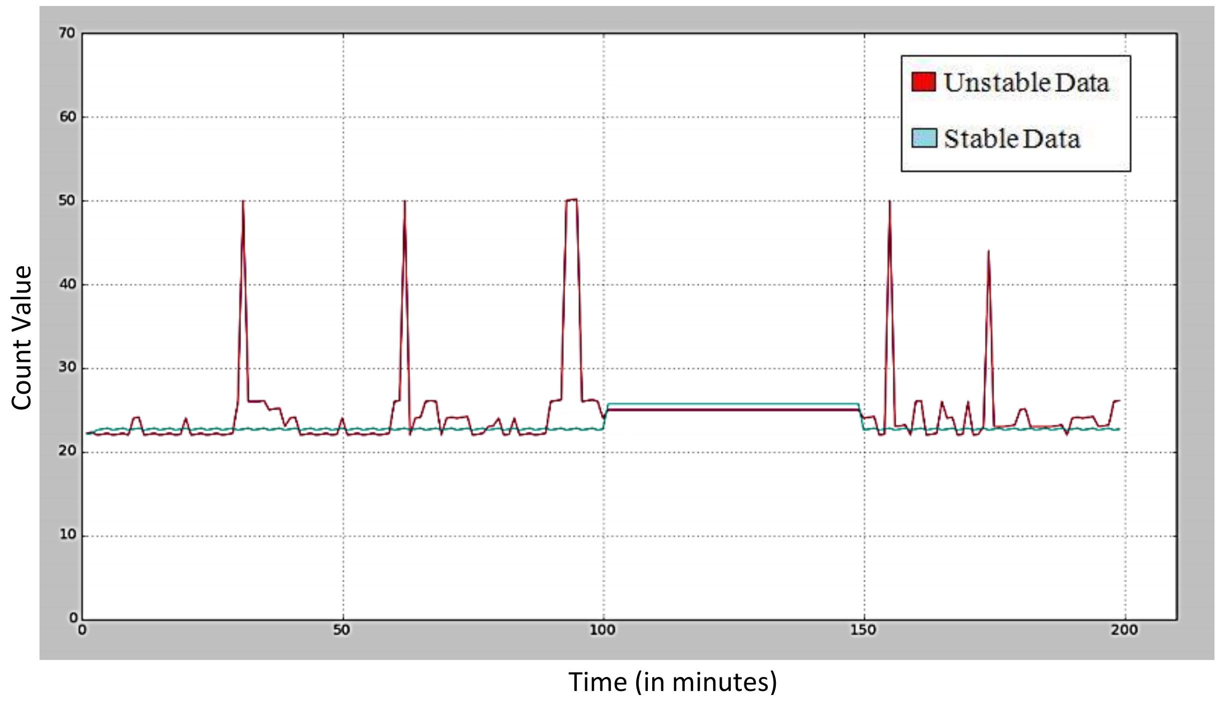


Figure 5.8. Temperature Sensor Data (Value)

Result:

The stabilized temperature values gradually increase and decrease despite suffering sudden spikes in temperature data. Such value only rises when peaks cross a particular threshold for a pre-defined span. This scenario is well depicted in Table 5.4 where unwanted data is tapered for normalize value.

Table 5.4: Temperature Sensor Data (Value)

Un-Stable temperature data on basis of Data value consisting of random voltage glitches and noise	**Stable temperature data on basis of Data value above particular threshold for a particular Time frame**
22.2	22.1
22.3	22.2
26.0	22.6
50.0	22.7
26.0	22.8
25.0	22.6
25.1	22.7
25.2	22.8
23.0	22.6
24.0	22.7
24.1	22.8
22.0	22.6
26.2	22.8
50.0	22.6
50.1	22.7
50.2	22.8
26.0	22.6
26.1	22.7
26.2	22.8
26.0	22.6
24.0	22.7
25.0	25.7
25.0	25.7
25.0	25.7
25.0	25.7
24.0	22.6
24.1	22.7
24.2	22.8

22.0	22.6
22.1	22.7
50.0	22.8
26.0	22.7
26.1	22.8
22.0	22.6
22.1	22.7
22.2	22.8
44.0	22.6
23.0	22.7
23.0	22.8

5.4 Stabilized Sensor Data Accessibility by "*IntSeN*" Android App

The sensor data which is stored in the cloud after cleaning and stabilizing process, could be displayed using Android application. This application provides the facility to access sensor data which comprise different varieties of sensor like temperature, pressure, and light.

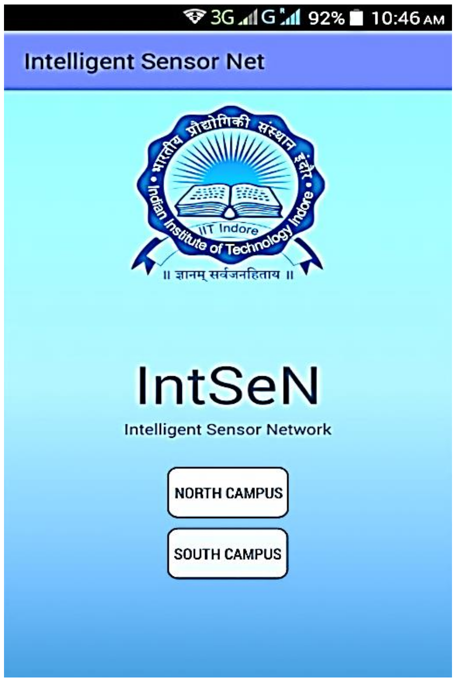


Figure 5.9. Main Dashboard for "*IntSeN*" Android App

The “*IntSeN*” Android App is designed to monitor environmental parameters and tested in premises of Indian Institute of Technology (IIT) Indore, Madhya Pradesh, India. Total 10 sensor motes are used to test the environmental parameters and are divided into two: North & South Campus at IIT Indore as shown in Figure 5.9.

Further, sensor data value could be accessed by selecting particular campus area. On selection, it would provide an option for choosing a type of sensor for display its associated value in that campus area. This scenario is well depicted in Figure 5.10.

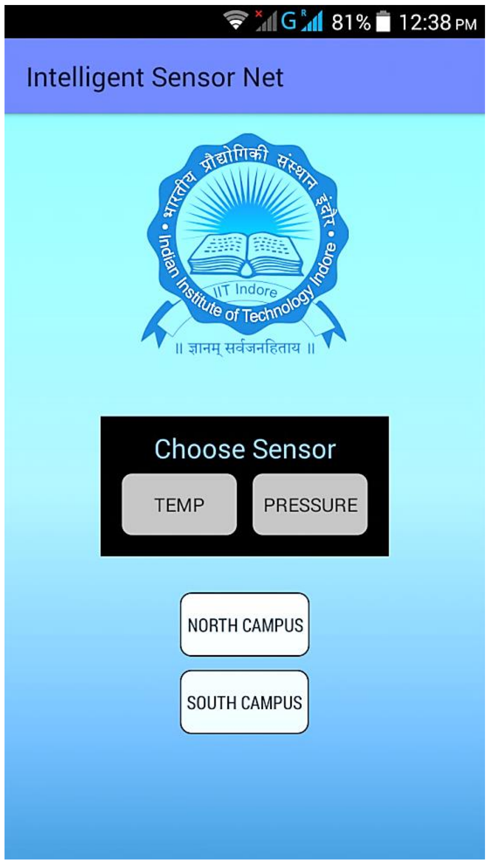


Figure 5.10. Choice for sensor type selection for two different Campus at IIT Indore

Once a particular sensor is selected, it would fetch sensor data from cloud where stabilized data is stored. This stabilization for sensor data does the filtration of noise in the data and such filtered data is stored at cloud location. This saves the unnecessary memory occupancy at cloud and is really a cost effecting model as cloud adhere the policy of '*pay-per-use*'.

When North Campus is select and afterward, a pressure sensor is chosen to display, then it fetch its associated data at run-time and this scenario is shown in Figure 5.11.

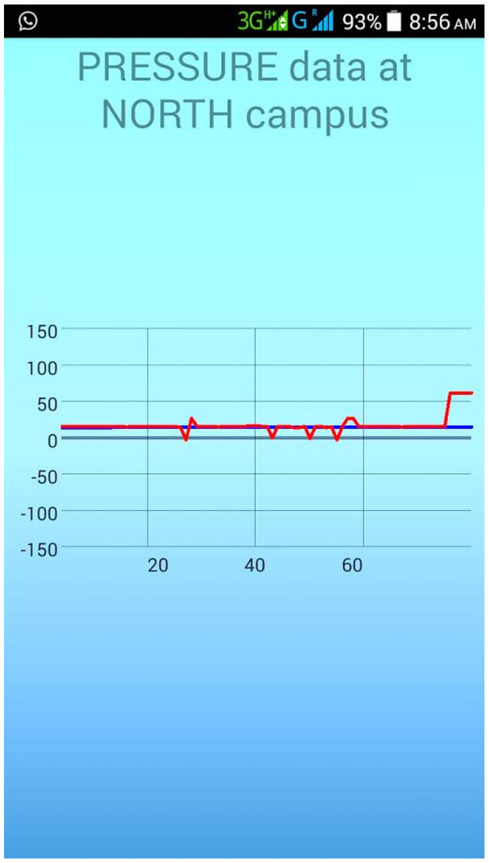


Figure 5.11.Pressure Data for North Campus at IIT Indore

5.5 Discussion

In this filtration process, data are taken by some rule. This data may have some error. In this section, the error has been analyzed. The well-known relative error is defined as follows.

$$\mathrm{Re}\,lative\ error = \frac{Absolute\,error}{True\,Value} \times 100$$

The true values are shown in 1st column and filtered value in the 2nd column in Table 5. If the received data is correct, then there must be an error. In the third column, the errors have been shown. If the error is small, then the data could be accepted. Otherwise, the data filtration may be taken as invalid.

Table 5.5: Relative error of filtered data

True value	**Filtered value**	**Relative error**
20.6	21.1	2.4%
20.1	21.1	4.9%
33	32.0	3.03%
31	32.0	3.22%
50.0	22.6	**54.8%**
50.1	22.7	**54.6%**

In this chapter, all the data are filtered using time tolerance, threshold cut, and category. These data are filtered using intelligent methods. But all the filtration could be useful if the error is less. In Table 5.5, last two rows show that the error is more than 50%.Thus these received data might be taken invalid or the process has some difficulty.

The sensor network is a collection of multiple motes which sensed the environmental characteristics and sends the data back to the repository through well-known interfaces. Such integration of embedded scheme like wireless sensor network with high-end systems like SOA system, need better interoperability, scalability, and reliability to implement this overall system. ESB extends the features of SOA which handles the actual data which is forwarded from sensor motes and stores at a local or global repository.

Sensed data may contain noise or error due to environmental hindrances and such noise needs to be reduced for stabilizing an overall system. This could be performed using data filtration and cleaning processes that could be done between ESB mediation level and sensor networks mote connection. This property makes any sensor network with an intelligent enough that could take a decision which needs to be filtered and cleaned. It also reduced the saving unwanted data directly to the repository (local or cloud).

Chapter 6

Conclusions and Scope for Future Work

6.1 Conclusions

The main objective of this thesis is to study and assess the data from different sensors that need to be stabilized and clean before keeping it at the cloud. The major contributions of this thesis include:

1) This thesis primarily discusses the detailed investigation for ESB and its core design and architectural principles, routing protocols, survey for existing ESBs available in the market. It has been analyzed that concept of ESB is to connect various service components together i.e. service containers. It has the capability to route messages/data more securely as compared to simple object access protocol (SOAP) service. ESB improves the architectural benefit associated with an enterprise solution for service integration and does organized service handling. By adopting ESB as their business solutions, one can reduce the risk that involved the data management issues and resultant would automatically increase the RoI.

2) This thesis also explains about architecture framework for ESB with its core principle for designing. Further, it highlights the importance of service stack with regard to delivering services in ESB. It also presents a comprehensive architecture view for ESB and what ESB actually comprises. It illustrates the concept of routing and mechanisms associated with message transformation. It also presents a brief discussion on the classification of routing techniques.

3) The semantic for service execution is also analyzed at global configuration with data acquisition. This details about mathematical

formulation which is derived from the relationship between data, multiple services and a storage at cloud. This model is also aimed at developing a novel approach for extending the features of SOA systems that supports multiple service components. This communication could be strengthened by well-defined '*link'* between correspondents '*Node'*. Here, '*Node'* could be a client or just sensor node which actually rises request for services and communication between them is governed by a 'link'. It also investigates the accessibility of dataset for two different levels; one at a local level (native server) and another at a global level (cloud servers).

4) Another contribution is stabilization of sensor data and such data is filtered using time tolerance, threshold cut, and segregation. Such data need to be cleaned using an intelligent method that improves the sensor data if error is detected. The sensor captured the environmental characteristics and sends the data to the global repository through a well-known interfaces. Such interfacing of embedded device and cloud with high-end systems could support better interoperability, scalability, and reliability to implement this overall service-oriented system.

5) While sensing data at sensor mote, it may suffer from severe fluctuations that results in the sudden voltage change. It may be due to losing connections, short circuits and link breakage problems that fluctuate the data value at particular sensor unit. This creates unstable, discontinuity and noisy data during capturing at sensor mote. This thesis proposed an "*intelligent sensor network (IntSeN)*" which is multi-level architecture for handling sensor data. It helps in stabilizing the annoying spike in the sensor data using fuzzy tolerance. This saves the unwanted data to get store at cloud server and reduce the data load.

6.2 Future Research Scope

In sensor application, a large number of services are integrating for data exchange. This could be achieved through a common platform that supports interoperability, scalability, and reusability for multiple services. Whenever, data is transmitted from one sensor node to another, there is always be a bit-error probability especially wireless communication. The work done in thesis could be extended in this regard where type of error checksum mechanism should verified based on parity bit (even or odd). This study should also focus on power (energy) consumption at sensor node.

Also, ESB act as a depletion layer that efficiently handles various overheads during communication. The specific business applications that are complex, dynamic and highly computational, requires a lot of effort in processing and requires somewhat more than general SOA implementation. This situation introduces a new concept called - '*Integration Suite*' that provides various integration methodologies for handling different domain specific applications. In future, this model could be implemented to Internet of Things (IoT) which is more scalable, efficient and robust for a different kind of sensing mechanism.

References

[1] J. Hu, F. E. Luo, J. Li, X. Tong, and G. Liao, “SOA-based Enterprise Service Bus”, IEEE International Symposium on Electronic Commerce and Security (ISECS), pp. 536-539, Guangzhou City, China, 3-5 August 2008.

[2] G. Li, J. Xiao, C. Li, S. Li, and J. Chen, “A Comparative Study between Soft System Bus and Enterprise Service Bus”, IEEE International Conference on Computer Science and Service System (CSSS) pp. 557-561, Nanjing, China, 11-13 August 2012.

[3] K. Ueno and M. Tatsubori, “Early Capacity Testing of an Enterprise Service Bus”, IEEE International Conference on Web Services (ICWS), pp. 709-716, Chicago, USA, 18-22 September 2006.

[4] M.T. Schmidt, “The Enterprise Service Bus: Making service-oriented architecture real”, IBM Systems Journal, Vol. 44, No.4, pp. 781-797, 2005.

[5] J. Cheng, “Persistent Computing Systems as Continuously Available, Reliable, and Secure Systems”, IEEE First International Conference on Availability, Reliability and Security (ARES), pp. 1-8, Vienna, Austria, 20-22 April 2006.

[6] M. Psiuk, T. Bujok, and K. Zielinski, “Enterprise Service Bus Monitoring Framework for SOA Systems”, IEEE Transactions on Services Computing, Vol. 5, No.3, pp. 450-466, 2012.

[7] A. M. Riad, A. E. Hussan, and Q. F. Hassan, “Design of SOA-based Grid Computing with Enterprise Service Bus”, International Journal on Advances in Information Sciences and Service Sciences, Vol. 2, No. 1, pp. 71-82, 2010.

[8] F. Ning, D. Junhua, and G. Yan, “A Service Composition Environment based on Enterprise Service Bus”, IEEE 11th Intl Conf on Ubiquitous Intelligence & Computing and IEEE 11th Intl Conf on Autonomic & Trusted Computing and IEEE 14th Intl Conf on Scalable Computing and Communications and Its Associated Workshops and IEEE 11th Intl Conf on Ubiquitous Intelligence & Computing and IEEE 11th Intl Conf on Autonomic & Trusted Computing and IEEE 14th Intl Conf on Scalable Computing and Communications and Associated Symposia/Workshops, pp. 738-743, 09-12 December 2014.

[9] Fiorano ESB. http://www.fiorano.com/ [Accessed on 21.05.17]

[10] Y. Liu, I. Gorton, and L. Zhu, “Performance Prediction of Service-Oriented Architecture based on an Enterprise Service bus”, 31st Annual International Computer Software and Application Conference, Vol. 1, pp. 327-334,Beijing, China, 24-27 July 2007.

[11] A. M. Bidgoli, and P. Nabhani, “Intelligent Conceptual Message Routing in Enterprise Service Bus (ESB)”, World Congress in Computer Science, Computer Engineering and Applied Computing, pp. 1-6, Los Vegas, USA, 18-21 July 2011.

[12] J. Wu, and X. Tao, “Research of Enterprise Application Integration Based on ESB”, 2nd International Conference on advanced

Computer Control (ICACC), pp. 90-93, Shenyang, China, 27-29 March 2010.

[13] G. A. Lewis, D. B. Smith, and K. Kontogiannis, “A Research Agenda for Service-Oriented Architecture (SOA): Maintenance and Evolution of Service-Oriented Systems”, [Master Dissertation] Software Engineering Institute, Carnegie Mellon University, 2010.

[14] K. Zielinnski, T. Szydło, R. Szymacha, J. Kosiński, J. Kosińska, and M. Jarzab, “Adaptive SOA Solution Stack”, IEEE Transactions on Services Computing, Vol. 5, No.2, pp. 149-163, 2012.

[15] D. Zmuda, M. Psiuk, and K. Zieliński, “Dynamic Monitoring Framework for the SOA Execution Environment”, Procedia Computer Science, Vol. 1, No. 1, pp. 125-133, 2010.

[16] M. Breest “An Introduction to the Enterprise Service Bus”, 2006. http://www.cin.ufpe.br/~in1062/intranet/bibliografia/fose-the_enterprise_service_bus.pdf [Accessed on 21.05.17]

[17] J. Yin, H. Chen, S. Deng, and Z. Wu, “A Dependable ESB Framework for Service Integration”, IEEE Internet Computing, Vol. 13, No. 2, pp. 26-34, 2009.

[18] C. A. Sun, E. el Khoury, and M. Aiello, “Transaction Management in Service-Oriented Systems: Requirements and a Proposal”, IEEE Transactions on Services Computing, Vol. 4, No. 2, pp. 167-180, 2011.

[19] S.P. Ahuja, and A. Patel, “Enterprise Service Bus: A Performance Evaluation”, Journal of Communications and Network, Vol. 3, No. 3, pp. 133-140, 2011.

[20] M. Chen, V.P.J. Chi, and H. C. Li, “An Enterprise Architecture Approach to Building a Service-Oriented Enterprise”, 6th International Conference on Service Systems and Service Management (ICSSSM), pp.704-709, Xiamen, China, 8-10 June 2009.

[21] S. Ortiz Jr., “Getting on Board the Enterprise Service Bus”, IEEE Computer Magazine, Vol. 40, No. 4, pp. 15-17, 2007.

[22] Aberdeen Group (USA). Enterprise Service Bus: An SOA Middleware Foundation. 2006. www.fiorano.com/docs/aberdeen.pdf

[23] P. Xu, and W. Du, “ERDSR: Efficient and Reliable Dynamic Service Routing in Enterprise Service Bus”, 3rd IEEE International Conference on Intelligent System Design and Engineering Applications (ISDEA), p. 712 – 715, Hong Kong, China, 16-18 January 2013.

[24] B. Wu, S. Liu, L. Wu, “Dynamic Reliable Service Routing in Enterprise Service Bus”, IEEE Asia-Pacific Services Computing Conference, pp. 349–354, Yilan, Taiwan, 09-12 December 2008.

[25] X. Bai, J. Xie, B. Chen, and S. Xiao, “DRESR: Dynamic Routing in Enterprise Service Bus”, IEEE International conference on e-Business Engineering (ICEBE), pp. 528–531, Hong Kong, China, 24-26 October 2007.

[26] D. Bo, D. Kun, and Z. Xiaoyi, “A High Performance Enterprise Service Bus Platform for Complex Event Processing”, 7th IEEE International Conference on Grid and Cooperative Computing, pp. 577–582, Shenzhen, China, 24-26 October 2008.

[27] P. Brebner, “Service-Oriented Performance Modeling the MULE Enterprise Service Bus (ESB) Loan Broker Application”, 35th IEEE Euro micro Conference on Software Engineering and Advanced Applications, pp. 404-411, Patras, Greece, 27-29 August2009.

[28] X. Ruzhi, B. Jin, and W. Yufei, “The Research and Implementation of Power Application System Integration Based on Enterprise Service Bus”, IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS), pp. 466 – 469, Xiamen, China, 29-31 October 2010.

[29] W. Huang, B. C. Sun, H. Zhao, and Y.S. Hu, “Enterprise Service Bus Based on OSGi”, 22nd International Conference on Industrial Engineering and Engineering Management, pp. 233-245, Wuhan, China, 12-14 October 2015.

[30] Sonic: Enterprise Service Bus. http://www.aurea.com/technology-solutions/enterprise-service-bus, 2013. [Accessed on 21.05.17]

[31] J. Ji-chen, and G. Ming, “Enterprise Service Bus and an Open Source Implementation”, IEEE International Conference on Management Science and Engineering (ICMSE), pp. 926 – 930, Lille, France, 5-7 October 2006.

[32] IBM Research Work, Understand Enterprise Service Bus scenarios and solutions in Service-Oriented Architecture. https://www.ibm.com/developerworks/library/ws-esbscen/ [Accessed on 21.05.17]

[33] IBM, “Exploring the Enterprise Service Bus: Discover how an ESB can help you meet the requirements for your SOA solution”, https://www.ibm.com/developerworks/library/ar-esbpat1/ [Accessed on 21.05.17]

[34] K. J. Lin, M. Panahi, Y. Zhang, J. Zhang, and S. H. Chang, “Building Accountability Middleware to Support Dependable SOA, IEEE Internet Computing, Vol. 13, No. 2, pp. 16-25, 2009.

[35] TIBCO ActiveMatrix® Service Bus. http://www.tibco.co.in/ [Accessed on 21.05.17]

[36] Window Azure Bus. http://azure.microsoft.com/en-us/services/service-bus/ [Accessed on 21.05.17]

[37] IBM Research Work. Exploring the Enterprise Service Bus: Part 4: Federated connectivity in the enterprise. http://www.ibm.com/developerworks/websphere/library/techarticles/0901_flurry/0901_flurry.html [Accessed on 21.05.17]

[38] X. Tang, S. Sun, X. Yuan, and D. Chen, “Automated Web Service Composition System on Enterprise Service Bus”, 3rd IEEE International Conference on Secure Software Integration and Reliability Improvement, pp. 9-13, Shanghai, China, 8-10 July 2009.

[39] WSO2 ESB. http://www.wso2.com/products/enterprise-service-bus/ [Accessed on 21.05.17]

[40] Adroit Logic Ultra ESB. http://www.adroitlogic.org/products/ultraesb.html [Accessed on 21.05.17]

[41] Integration Pattern Overview. http://www.eaipatterns.com/eaipatterns.html [Accessed on 21.05.17]

[42] O. Harcuba, and P. Vrba, "Unified REST API for supporting the semantic integration in the ESB-based architecture", IEEE International Conference on Industrial Technology (ICIT), pp. 3000-3005, Seville, Spain, 17-19 March 2015.

[43] R. Benosman, K. Barkaoui, and Y. Albrieux, "Design and implementation of a massively parallel ESB", 11th International Symposium on Programming and Systems (ISPS), pp. 89-95, Algiers, Algeria, 22-24 April 2013.

[44] K. Shi, F. Gao, Q. Xu, and G. Xu, "Integration framework with semantic aspect of heterogeneous system based on ontology and ESB", 26th Chinese Control and Decision Conference (2014 CCDC), pp.4143-4148, Changsha, China, 31 May -02 June 2014.

[45] Red Hat JBoss ESB. http://www.jbossesb.jboss.org/ [Accessed on 21.05.17]

[46] Talend Enterprise ESB. http://www.talend.com/products/esb [Accessed on 21.05.17]

[47] Oracle ESB. http://www.oracle.com/technetwork/middleware/ service-bus/overview /index.html [Accessed on 21.05.17]

[48] M. Fowler, “Micro Services Architecture - Death of Enterprise Service Bus (ESB)”, Thought Work Incorporation, USA. http://www.kai-waehner.de/blog/2015/01/08/micro-services-architecture-death-enterprise-service-bus-esb/ [Accessed on 21.05.17]

[49] Apache Synapse ESB. Retrieved From: https://synapse.apache.org/ [Accessed on 21.05.17]

[50] A. Ferreira, A. Pereira, N. Rodrigues, J. Barbosa, and P. Leitao, “Integration of an agent-based strategic planner in an enterprise service bus ecosystem”, 13th IEEE International Conference on Proceedings of Industrial Informatics (INDIN), pp. 1336-1341, Cambridge, UK, 22-24 July 2015.

[51] N. RajKumar and V. Vinod, “Integrated Educational Information Systems for Disabled Schools via a Service Bus using SOA”, Indian Journal of Science and Technology, Vol. 8, No. 13, pp. 1-7, 2015.

[52] S. Kumari, and S. K. Rath, “Performance comparison of SOAP and REST based Web Services for Enterprise Application Integration”, IEEE International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 1656-1660, Kochi, India 10-13 August 2015.

[53] L. González, J. L. Laborde, M. Galnares, M. Fenoglio, and R. Ruggia, “An adaptive enterprise service bus infrastructure for service based systems”, Springer Service-Oriented Computing (ICSOC) Workshops, pp. 480-491, Berlin, Germany, 2-5 December 2014.

[54] G. Pan, L. Zhang, Z. Wu, S. Li, L. Yang, M. Lin, and Y. Shi, "Pervasive Service Bus: Smart SOA Infrastructure for Ambient Intelligence", IEEE Intelligent Systems, Vol. 29, No. 4, pp. 52-60, 2014.

[55] N. Ulltveit-Moe, and V. Oleshchuk, "A novel policy-driven reversible anonymisation scheme for XML-based services", Information System, Vol. 48, pp. 164-178, 2015.

[56] K. A. Alam, R. Ahmad, A. Akhunzada, M. H. N. Md Nasira, and S. U. Khan, "Impact analysis and change propagation in service-oriented enterprises: A systematic review", Elsevier Information System, Vol. 54, pp. 43-73, 2015.

[57] L. Garces-Erice, "Building an enterprise service bus for real-time SOA: a messaging middleware stack", 33rd Annual International Computer Software and Applications Conference (COMPSAC'09), Vol. 2, pp. 79-84, 2009.

[58] W. Roshen, "Enterprise service bus with USB-like universal ports", Ninth IEEE European Conference on Web Services (ECOWS), pp. 177-183, Lugano, Switzerland, 14-16 September 2011.

[59] T. Hedlund, "Design and Proof-of-Concept Implementation of Proxy-based Stream Handling for an Enterprise Service Bus", Institutionen för datavetenskap, [Master Dissertation] Department of Computer and Information Science, Linköpings universitet, Sweden, 2014.

[60] S. Subashini, and V. Kavitha, “A survey on security issues in service delivery models of cloud computing”, Journal of Network and Computer Applications, Vol. 34, No. 1, pp. 1-11, 2011.

[61] L. J. Zhang, “Web Services Research for Emerging Applications: Discoveries and Trends: Discoveries and Trends”, IGI Global, USA, 2010.

[62] J. Cámara, A. Lopes, D. Garlan, and B. Schmerl, “Adaptation impact and environment models for architecture-based self-adaptive systems”, Science of Computer Programming, Volume 127, pp. 50–75, 2016.

[63] T. Klaus, J. E. Blanton, and S. C. Wingreen, “User Resistance Behaviors and Management Strategies in IT-Enabled Change”, Journal of Organizational and End User Computing, Vol. 27, No.1, pp. 57-76, 2015.

[64] J. L. Fiadeiro, A. Lopes, and J. Abreu, “A formal model for service-oriented interactions”, Science of Computer Programming, Vol. 77, No. 5, pp. 577-608, 2012.

[65] J. L. Fiadeiro, and A. Lopes, “A model for dynamic reconfiguration in service-oriented architectures”, Software & Systems Modeling, Vol. 12, No. 2, pp. 349-367, 2013.

[66] E. Kanagaraj, L.M. Kamarudin, A. Zakaria, R. Gunasagaran, and A. Y. M. Shakaff, “Cloud-based remote environmental monitoring system with distributed WSN weather stations”, IEEE conference on SENSORS, pp. 1-4, Busan, South Korea, 1-4 November 2015.

[67] I. Sims, C. O'Leary, and P. Boolaky, “Overreliance on Mathematical Accuracy of Computer Output: An Issue for IT

Educators" Journal of Organizational and End User Computing, Vol. 26, No. 3, pp. 47-64, 2014.

[68] P. T. Monteiro, E. Dumas, B. Besson, R. Mateescu, M. Page, A. T. Freitas, and H. De Jong, "A service-oriented architecture for integrating the modeling and formal verification of genetic regulatory networks", BMC bioinformatics, Vol. 10, Vol. 1, pp. 1-12, 2009.

[69] Y. Zhang, D. Zhang, M.M. Hassan, A. Alamri, and L. Peng, "CADRE: Cloud-assisted drug recommendation service for online pharmacies", Mobile Networks and Applications, Vol. 20, No. 3, pp. 348-355, 2015.

[70] S. Walther, and H. Wehrheim, "On-the-fly construction of provably correct service compositions–templates and proofs", Science of Computer Programming, Vol. 127, pp. 2-23, 2016.

[71] K. Liu, "Pragmatic computing–a semiotic perspective to web services", Springer International Conference on E-Business and Telecommunications, pp. 3-15, Barcelona, Spain, 28-31 July 2007.

[72] B. A. Myers, S. Y. Jeong, Y. Xie, J. Beaton, J. Stylos, R. Ehret, J. Karstens, A. Efeoglu, and D. K. Busse, "Studying the documentation of an API for enterprise Service-Oriented Architecture", Journal of Organizational and End User Computing, Vol. 22, No.1, pp. 23-51, 2010.

[73] M. F. Gholami, M. Sharifi, and P. Jamshidi, "Enhancing the OPEN Process Framework with service-oriented method fragments", Software & Systems Modeling, Vol. 13, No. 1, pp. 361-390, 2014.

[74] R. Kyusakov, J. Eliasson, J. Delsing, J. van Deventer, and J. Gustafsson, “Integration of wireless sensor and actuator nodes with IT infrastructure using service-oriented architecture”, IEEE Transactions on industrial informatics, Vol. 9, No. 1, pp. 43-51, 2013.

[75] R. Qumsiyeh, and Y. K. Ng. “Enhancing web search by using query-based clusters and multi-document summaries”, Knowledge and Information Systems, Vol. 47, No. 2, pp. 355-380, 2016.

[76] M. S. Hossain, and G. Muhammad, “Cloud-assisted speech and face recognition framework for health monitoring”, Mobile Networks and Applications, Vol. 20, No. 3, pp. 391-399, 2015.

[77] X. Liu, Y. Ma, G. Huang, J. Zhao, H. Mei, and Y. Liu, “Data-Driven Composition for Service-Oriented Situational Web Applications”, IEEE Transactions on Services Computing, Vol. 8, No. 1, pp. 2-16, 2015.

[78] P. Reinecke, K. Wolter, and M. Malek, “A survey on fault-models for QoS studies of service-oriented systems”, [PhD dissertation] Humboldt-Universit¨at zu, Berlin, Germany, 2010.

[79] S. Capelli, and P. Scandurra, “A framework for early design and prototyping of service-oriented applications with design patterns”, Computer Languages, Systems & Structures, Vol. 46, pp. 140-166, 2016.

[80] B. Phillips, “Information technology management practice: impacts upon effectiveness”, Journal of Organizational and End User Computing, Vol. 25, No. 4, pp. 50-74, 2013.

[81] A. Alwadain, E. Fielt, A. Korthaus, and M. Rosemann, “Empirical insights into the development of a service-oriented enterprise architecture”, Data & Knowledge Engineering, Vol. 105, 39-52, 2016.

[82] Organization for the Advancement of Structured Information Standards (OASIS). https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf [Accessed on 21.05.17]

[83] J. Leguay, M. Lopez-Ramos, K. Jean-Marie, and V. Conan, “An efficient service oriented architecture for heterogeneous and dynamic wireless sensor networks”, IEEE 33rd Conference on Local Computer Networks (LCN), pp. 740-747, Montreal, Canada, 14-17 October 2008.

[84] O. Alhabashneh, R. Iqbal, F. Doctor, and A. James, “Fuzzy Rule Based Profiling Approach For Enterprise Information Seeking and Retrieval”, Information Sciences, Vol. 394-395, pp. 18-37, 2017.

[85] G.F. Anastasi, E. Bini, A. Romano, and G. Lipari, “A service-oriented architecture for QoS configuration and management of wireless sensor networks”, IEEE Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1-8, Bilbao, Spain, 13-16 September 2010.

[86] Y. Sahni, J. Cao, and X. Liu, “MidSHM: A Middleware for WSN-based SHM Application using Service-Oriented Architecture”, Future Generation Computer Systems, Vol. 80, pp. 263-274, March 2018.

[87] M. S. Familiar, J. F. Martínez, and L. López, “Pervasive smart spaces and environments: a service-oriented middleware architecture for wireless ad hoc and sensor networks”, International Journal of Distributed Sensor Networks, pp. 1-11, 2012.

[88] T. Schär, “Adaptive RESTful Architecture for Wireless Sensor Networks”, [Master Thesis] Pervasive & Artificial Intelligence Research Group, University of Fribourg, Switzerland, 2013.

[89] E. K. Ozorhan, E. K. Kuban, and N. K. Cicekli, “Automated composition of web services with the abductive event calculus”, Information Sciences, Vol. 180, No. 19, pp. 3589-3613, 2010.

[90] G. Moritz, F. Golatowski, C. Lerche, and D. Timmermann, “Beyond 6LoWPAN: Web services in wireless sensor networks”, IEEE Transactions on Industrial Informatics, Vol. 9, No. 4, pp. 1795-1805, 2013.

[91] N. L. Martínez, J. F. Martínez, and V. Hernández Díaz, “Virtualization of event sources in wireless sensor networks for the Internet of Things”, Sensors, Vol. 14, No. 12, pp. 22737-22753, 2014.

[92] X. Su, L. Wu, and P. Shi, “Sensor networks with random link failures: distributed filtering for T–S fuzzy systems”, IEEE Transactions on Industrial Informatics, Vol. 9, No. 3, pp. 1739-1750, 2013.

[93] F. Zhang, Z. Zhou, Q. Liu, and W. Xu, “An intelligent service matching method for mechanical equipment condition monitoring

using the fibre Bragg grating sensor network", Enterprise Information Systems, Vol. 11, No. 2, pp. 284-309, 2017.

[94] W. T. Lee, and S. P. Ma, "Process modeling and analysis of service-oriented architecture–based wireless sensor network applications using multiple-domain matrix", International Journal of Distributed Sensor Networks, Vol. 12, No. 11, pp. 1-15, 2016.

[95] L. Stoimenov, M. Bogdanovic, and S. Bogdanovic-Dinic, "ESB-based sensor web integration for the prediction of electric power supply system vulnerability", Sensors, Vol. 13, No. 8, pp. 10623-10658, 2013.

[96] M. Eisenhauer, P. Rosengren, and P. Antolin, "A development platform for integrating wireless devices and sensors into ambient intelligence systems", IEEE 6th Annual workshop on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), pp. 1-3, Rome, Italy, 22-26 June 2009.

[97] R. S. Alonso, D. I. Tapia, J. Bajo, Ó. García, J. F. de Paz, and J. M. Corchado, "Implementing a hardware-embedded reactive agents platform based on a service-oriented architecture over heterogeneous wireless sensor networks", Ad Hoc Networks, Vol. 11, No.1, pp. 151-166, 2013.

[98] S. de Deugd, R. Carroll, K. Kelly, B. Millett, and J. Ricker, "SODA: Service oriented device architecture. IEEE Pervasive Computing", Vol. 5, No. 3, pp. 94-96, 2006.

[99] D. Romero, and F. Vernadat, “Enterprise information systems state of the art: Past, present and future trends. Computers in Industry, Vol. 79, pp. 3-13. 2016.

[100] X. Qiu, H. Luo, G. Xu, R. Zhong, and G. Q. Huang, “Physical assets and service sharing for IoT-enabled Supply Hub in Industrial Park (SHIP)”, International Journal of Production Economics, Vol. 159, pp. 4-15, 2015.

[101] C. Chang, S. N. Srirama, and R. Buyya, “Mobile Cloud Business Process Management System for the Internet of Things: Review, Challenges and Blueprint”. arXiv preprint arXiv:1512.07199, 2015.

[102] M. A. Martínez-Carreras, F. J. García Jimenez, and A. F. Gómez Skarmeta, “N.L, Enterprise Information Systems, Vol. 9, No. 4, pp. 401-435, 2015.

[103] R. Alena, J. Ossenfort, T. Stone, and J. Baldwin, “Wireless Space Plug-and-Play Architecture (SPA-Z)”, IEEE Aerospace Conference, pp. 1-17, Big Sky, MT, USA, 1-8 March 2014.

[104] F. Palumbo, J. Ullberg, A. Štimec, F. Furfari, L. Karlsson, and S. Coradeschi, “Sensor network infrastructure for a home care monitoring system”, Sensors, Vol. 14, No. 3, pp. 3833-3860, 2014.

[105] P. Castillejo, J.F. Martínez, L. López, and G. Rubio, “An internet of things approach for managing smart services provided by wearable devices”, International Journal of Distributed Sensor Networks, Vol. 9, Issue: 2, pp. 1-9, 2013.

[106] D. Rocchini, G. M. Foody, H. Nagendra, C. Ricotta, M. Anand, K.S. He, V. Amicih, B. Kleinschmiti, M. Försteri, S. Schmidtlein and H. Feilhauer, "Uncertainty in ecosystem mapping by remote sensing", Computers & Geosciences, Vol. 50, pp. 128-135, 2013.

[107] Medjahed, H., Istrate, D., Boudy, J., Baldinger, J. L., & Dorizzi, B. "A pervasive multi-sensor data fusion for smart home healthcare monitoring", IEEE International Conference on Fuzzy Systems (FUZZ), pp. 1466-1473, Taipei, Taiwan, 27-30 June 2011.

[108] W. Oyomno, P. Jäppinen, and E. Kerttula, "Privacy implications of context-aware services", ACM 4th International ICST conference on communication system software and middleware, pp. 1-7, Dublin, Ireland, 15-19 June 2009.

[109] L. Sánchez, I. Couso, and J. Casillas, "Genetic learning of fuzzy rules based on low quality data", Fuzzy Sets and Systems, Vol. 160, No. 17, pp. 2524-2552, 2009.

[110] P. Jamshidi, S. Khoshnevis, R. Teimourzadegan, A. Nikravesh, and F. Shams, "Toward automatic transformation of enterprise business model to service model", ICSE Workshop on Principles of Engineering Service Oriented Systems (PESOS), pp. 70-74, Vancouver, BC, Canada, 18-19 May 2009.

[111] A. Dan, R. Johnson, and A. Arsanjani, "Information as a service: Modeling and realization", IEEE International workshop on Systems development in SOA environments (SDSOA), pp. 1-6, Minneapolis, USA, 20-26 May 2007.

[112] U. C. Benz, P. Hofmann, G. Willhauck, I. Lingenfelder, and M. Heynen, “Multi-resolution, object-oriented fuzzy analysis of remote sensing data for GIS-ready information”, ISPRS Journal of photogrammetry and remote sensing, Vol. 58, No. 3, pp. 239-258, 2004.

[113] G. Mauris, V. Lasserre, and L. Foulloy, “Fuzzy modeling of measurement data acquired from physical sensors”, IEEE Transactions on Instrumentation and Measurement, Vol. 49, No. 6, pp. 1201-1205, 2000.

[114] R. Lu, H. Zhu, X. Liu, J. K. Liu, and J. Shao “Toward efficient and privacy-preserving computing in big data era”, IEEE Network, Vol. 28, No. 4, pp. 46-50, 2014.

[115] K. A. Delin, “Sensor Webs in the Wild”, Wireless Sensor Networks: A Systems Perspective, Artech House, 2005.

LIST OF PUBLICATIONS

<u>Publications from thesis</u>

Journal papers:-

1. R. S. Bhadoria, N. S. Chaudhari, G. S. Tomar, "The Performance Metric for Enterprise Service Bus (ESB) in SOA System: theoretical underpinnings and empirical illustrations or information processing", Information Systems (***Elsevier***), Vol. 65, pp. 158-171, 2016. {**Impact Factor: 2.777**}

2. R. S. Bhadoria, N. S. Chaudhari, S. Samanta, "Uncertainty in Sensor Data Acquisition for SOA System", Neural Computing and Applications (***Springer***), February 2017. ***DOI***: 10.1007/s00521-017-2910-2 {**Impact Factor: 2.505**}

3. R. S. Bhadoria, N. S. Chaudhari, "Pragmatic Sensory Data Semantics with Service-Oriented Computing", Journal of Organisational and End User Computing (***IGI Global***, ***USA***), March 2017. [*Accepted*] {**Impact Factor: 0.759**}

4. R. S. Bhadoria, N. S. Chaudhari, T. N. Vidanagama "Analyzing the Role of Interfaces in Enterprise Service Bus: A Middleware Epitome for Service-Oriented Systems", Computer Standards and Interface (***Elsevier***), Vol. 55, pp. 146-155, January 2018 {**Impact Factor: 1.633**}

5. R. S. Bhadoria, "Performance Analysis for Enterprise Service Bus in SOA System", International Journal of IT business Strategy Management, Vol. 1, No.1, 2015. {**Impact Factor: N.A.**}

6. R. S. Bhadoria, N. S. Chaudhari, "An Intelligent Sensor Network (IntSeN): A Service-Oriented System for Stabilized Sensor Data", **IEEE** Transactions on Services Computing [*Under Review*] {**Impact Factor: 3.520**}

Conference papers:-

1. R. S. Bhadoria, N. S. Chaudhari, “Provisioning for Sensor Data using Enterprise Service Bus: A Middleware Epitome”, Springer scientific international conference on safety & security (SICC 2017), Rome, Italy, June 2017.

2. N. Chaudhari, R. S. Bhadoria, S. Prasad , “Information Handling and Processing using Enterprise Service Bus in Service-Oriented Architecture System”, 7th *IEEE* International Conference on Computation Intelligence & Computer Network (CICN 2016), Tehri, Uttarakhand, India, December 2016.

Publications apart from thesis

Book Chapters:-

1. R. S. Bhadoria, “Security Architecture for Cloud Computing”, In Handbook of Research on Securing Cloud-Based Databases with Biometric Applications, Eds. G.C. Deka and S. Bakshi, IGI Global Inc. (USA),pp. 47-71, 2015.

2. R. S. Bhadoria, “Performance of Enterprise Architecture in Utility Computing”, In Handbook of Research on Emerging Research Surrounding Power Consumption and Performance Issues in Utility Computing, Eds. G. C. Deka, G.M. Siddesh, K. G. Srinivasa and L.M. Patnaik , IGI Global Inc. (USA), 2015.

Conference paper:-

1. R. Yadav, R. S. Bhadoria, “Performance Analysis for Android Runtimes Environment”, 6th IEEE International Conference on Communication System & Network Technologies (CSNT 2015), Gwalior, India, April 2015.