# A Lightweight, Reliable, Secure, Paginated, Disconnected, and Distributed Message Transaction Model for Wireless Mobile Environment

A Thesis Submitted to the University of Calicut in Partial Fulfilment of the Requirements for Award of the Degree of

**Doctor of Philosophy**

in Computer Science, under Faculty of Science
by

**Mohammed Shameer M C**
**(Reg. No: FKAPRSE001)**

Under the guidance of
**Dr. Abdul Haleem P P**

**DEPARTMENT OF COMPUTER SCIENCE**
**FAROOK COLLEGE (AUTONOMOUS)**

July 2024

I would like to dedicate this thesis to those whom I love

# Acknowledgements

In the name of Allaah, the Most Gracious and the Most Merciful. The first and foremost, I would thank The Almighty for bestowing me in all realms of my life.

I am deeply grateful to my mentor and supervisor Dr. Abdul Haleem P P, Associate Professor, Department of Information Technology, Kannur University. For nearly eight years, he has guided, supported and trusted me. He invested his time and energy for the completion of my work in a better way. His timely suggestion opened up new avenues in the research progress. His involvements are really influential in shaping my experiment methods. His motivation and guidance was my strength in weak hours.

The Research Department of Computer Science, Farook College (Autonomous) provided a stimulating and encouraging environment for carrying out this research work. I am thankful to the Principal, Dr. Aysha Swapna K. A. and former principals, Prof. E. P. Imbichikoya and Dr. K. M. Naseer, for their constant support and keen interest bestowed on me. I also appreciate all the support I have received from Dr. V. Kabeer, HOD, Department of Computer Science, Farook College (Autonomous).

I extend my gratitude to my friends and colleagues in the Department and outside for their constant support, constructive suggestions and criticisms of my work and I cherish the time spent together in discussing them, which I really consider as fruitful.

I wish to thank the reviewers of this dissertation, editors of journals, subject experts, mentors and other well wishers for their contributions.

Finally, I would like to express my gratitude to my parents, wife and children for their tremendous understanding and constant encouragement; no matter what choices I have made, they supported me. It would be impossible for me to complete my work without their wonderful understanding attitude.

Mohammed Shameer M C

# Declaration

I hereby certify that the thesis titled “A Lightweight, Reliable, Secure, Paginated, Disconnected, and Distributed Message Transaction Model for Wireless Mobile Environment” submitted to the University of Calicut for the award of the degree of Doctor of Philosophy in Computer Science under the Faculty of Science, is an independent work done by me under the guidance and supervision of Dr. Abdul Haleem P P, Associate Professor, Department of Information Technology, Kannur University, Kerala. I also declare that this thesis contains no material which has been previously submitted for the award of any other degree, diploma, associateship, or fellowship of any university or other institutions of higher studies, to the best of my knowledge and belief. I also declare that this thesis does not include any materials previously published by any other person, except where due reference is made in the text.

(Signature)

**Date**:

**Place**: Farook College

**Name**: Mohammed Shameer M C

**Reg. No**: FKAPRSE001

# Declaration

I hereby certify that the thesis titled "A Lightweight, Reliable, Secure, Paginated, Disconnected, and Distributed Message Transaction Model for Wireless Mobile Environment" submitted by Mr. Mohammed Shameer M C (Reg. No: FKAPRSE001), for the award of the Degree of Doctor of Philosophy in Computer Science, to the University of Calicut during the period of his study at Farook College (Autonomous), Farook College P. O., Kozhikode 673632, affiliated to the University of Calicut, is a bonafide record of original research work carried out by him under my guidance and supervision. It is his independent work and has not been included in any other thesis submitted previously for the award of any degree, diploma, associateship or fellowship of any other university or institutions of higher studies. I further certify that the modifications and recommendations suggested by the adjudicators are incorporated to the thesis. The soft copy attached is the same as that of the revised thesis. The contents of the thesis have been checked using an anti-plagiarism database and no unacceptable similarity was found through the software. The thesis is submitted as such to the University of Calicut with reference to the letter No. 278696/RESEARCH-C-ASST-1/2023/Admn dated 04.07.2024.

(Signature)

**Date**:

**Place**: Farook College

Dr. Abdul Haleem P P
Associate Professor
Dept. of Information Technology
Kannur University

# Abstract

LXML is a hierarchical data format inheriting XML features like extensibility, human readability, and schema awareness. Unlike XML, it uses level numbers instead of user-defined tags. This new format improves performance in verbosity, content density, parsing, serialization, deserialization, marshalling, unmarshalling, and transmission. LXML reduces message size by 40 - 48% compared to XML and 35 - 55% compared to JSON. Its schema is 80 - 90% smaller than XML Schema, offering substantial performance benefits over XML and JSON.

The novel transaction model could be realized in a distributed environment through a three-layer architecture (client agent, middleware, and host), supports transactions in disconnected and distributed wireless mobile environments. It integrates the efficient LXML message format with Service Oriented Architecture (SOA) to enhance transmission efficiency. Reliability is ensured through packet acknowledgement, sequencing mechanisms, and a message queuing system for network unavailability. The middleware offloads resource-intensive computations to a server. Performance evaluations show the model's superiority in upload and download times, LXML-to-web service conversions, transmission time, and content density, proving its suitability for modern wireless mobile environments.

To counter rewriting attacks, a three-layer architecture safeguards message transmissions using LXML Schema. This involves adding four fields (checksum, timestamp, child count, and digital signature) and encrypting messages. Case studies demonstrate the approach's effectiveness in detecting and preventing various rewriting attacks. An evaluation and analysis for the additional processing and memory overhead needed to implement these security measures reveals that they incur minimal additional processing time and memory. Consequently, the proposed lightweight, reliable, secure, paginated, disconnected, and distributed transaction model not only ensures reliability but guarantees the secure transmission of messages in an ever-evolving digital landscape as well.

In summary, the proposed innovative three-layer transaction model significantly advances beyond traditional XML and JSON, delivering exceptional efficiency, scalability, and security. This evolution is set to transform data exchange practices across diverse domains, from enterprise systems to the Internet of Things, heralding a more efficient and secure digital future.

*Keywords*: Data Interchange Formats, LXML, Transaction Model, Constrained Mobile Networks, Security, Rewriting Attacks.

## അബ്സ്ട്രാക്ട്

പ്രോസസ്സിംഗ് വേഗത, ബാറ്ററി ലൈഫ്, സംഭരണശേഷി എന്നിവയിൽ വയർലസ് മൊബൈൽ നെറ്റ്‌വർക്കുകൾക്കുള്ള സ്വാഭാവിക  പരിമിതികൾമൂലം സാധാരണ ഡാറ്റ ഇൻറ്റർചേഞ്ച്  മോഡലുകൾ കാര്യക്ഷമമായി ഉപയോഗിക്കുന്നത് പ്രായോഗികമല്ല. അതുകൊണ്ടുതന്നെ വയർലസ് മൊബൈൽ നെറ്റ്‌വർക്കുകളുടെ പരിമിതിയിൽ നിന്നുകൊണ്ട് മികച്ച കാര്യക്ഷമത ഉറപ്പുതരുന്ന പുതിയ ഡാറ്റ ഇൻറ്റർചേഞ്ച് മോഡലുകൾ വികസിപ്പിക്കേണ്ടത് ഒരു അനിവാര്യതയാണ്.

കുറഞ്ഞ വിഭവലഭ്യതയിൽ ഉയർന്ന വിശ്വസ്തതയും ഡാറ്റ സുരക്ഷിതത്വവും ഉറപ്പുനൽകുന്ന  LXML അടിസ്ഥാനമാക്കി ഒരു ഡിസ്കണക്റ്റഡ്   ട്രാൻസാക്ഷൻ മോഡൽ അവതരിപ്പിക്കുക എന്നതാണ് ഈ പ്രബന്ധത്തിൻറെ ഉദ്ദേശം. XML എന്ന ഡാറ്റ ഇൻറ്റർചേഞ്ച് രീതിയെ അടിസ്ഥാനപ്പെടുത്തി വികസിപ്പിച്ചെടുത്ത ഈ മോഡൽ, XML നിന്നും പലകാര്യങ്ങളിലും വിഭിന്നവും XML നെക്കാൾ വേഗതയിൽ ഡാറ്റ കൈമാറ്റം ചെയ്യാൻ പ്രാപ്തവുമാണ്. XML ഉപയോഗിക്കുന്ന ഉപഭോക്തനിർമ്മിത ടാഗുകൾക്ക് പകരം ലെവൽ നമ്പർ അനുസരിച്ചു ഡാറ്റ ക്രമീകരിക്കുകയാണ്  ഇവിടെ ചെയ്യുന്നത്. XML, JSON എന്നീ ഡാറ്റ കൈമാറ്റ രീതിയുമായി താരതമ്യം ചെയുമ്പോൾ LXMLന് യഥാക്രമം 40-48% വരെയും 35-55% വരെയും ഡാറ്റ വ്യാപ്തം കുറയ്ക്കാൻ കഴിയുമെന്ന് പരീക്ഷണങ്ങളിലൂടെ വെളിപ്പെട്ടതാണ്. കൂടാതെ ഡാറ്റ സാന്ദ്രത, പാഴ്‌സിങ് സമയം, സീരിയലൈസേഷൻ & ഡീസീരിയലൈസേഷൻ, മാർഷെല്ലിംഗ് & അൺമാർഷെല്ലിംഗ് സമയം എന്നി പ്രകടന മാനദണ്ഡങ്ങളിലും LXML മറ്റു മോഡലുകളെക്കാൾ വ്യക്തമായി മികച്ചുനിൽക്കുന്നു.

LXML സർവീസ് ഓറിയന്റെഡ്  ആർക്കിടെക്ചർ (SOA) യുമായി ബന്ധിപ്പിച്ചാൽ മികച്ച പ്രസരണസാമർഥ്യം പ്രകടമാകുന്നതാണ്. ഒരു ഡിസ്ട്രിബ്യൂട്ടഡ് സാങ്കേതിക ഘടനയിൽ മൂന്ന് പ്രവർത്തന പാളികളായാണ് ഈ നൂതന ഡാറ്റ ഇൻറ്റർചേഞ്ച് രീതി പ്രാവർത്തകമാക്കിയിരിക്കുന്നത്. നെറ്റ്‌വർക്ക് കണക്ഷൻ ലഭ്യമല്ലാത്ത സാഹചര്യങ്ങളിൽ, മെസ്സേജുകൾ ലക്ഷ്യസ്ഥാനത്തു  എത്തിക്കുന്നതിനു വേണ്ടിയും ഡാറ്റ കൈമാറ്റതുടർച്ച നിലനിർത്തുന്നതിനും മെസ്സേജ് ക്യൂ എന്ന സാങ്കേതികവിദ്യ വഴി ഈ മോഡലിന് സാധിക്കുന്നു. സംപ്രേക്ഷണ സമയം, അപ്‌ലോഡ്, ഡൗൺലോഡ് സമയം, ഉള്ളടക്കസാന്ദ്രത (Content Density) മുതലായ പ്രകടനസൂചകങ്ങൾ ഉൾപ്പെടുത്തി നടത്തിയ മൂല്യനിർണയ പരീക്ഷണങ്ങളിൽ കിട്ടിയ ഫലങ്ങൾ LXML അടിസ്ഥാനമാക്കിയ ട്രാൻസാക്ഷൻ  മോഡലിന്റെ  ശ്രേഷ്ഠത സൂചിപ്പിക്കുന്നു.

LXML ഒരു XML അടിസ്ഥിത പ്രസരണ മോഡലായതിനാൽ റീറൈറ്റിങ് ആക്രമണങ്ങൾക്ക് (rewriting attacks) വിധേയപ്പെടാൻ സാധ്യതയുണ്ട്. ആയതിനാൽ അനധികൃതവും കണ്ടെത്താനാകാത്തതുമായ കൃത്രിമത്വം തടയുന്നതിന് ഒരു മൂന്നു ലയർ സുരക്ഷ ചട്ടക്കൂട് (three layer security framework) ഇവിടെ മുന്നോട്ടുവെക്കുന്നു. LXML സ്‌കീമ,   എൻക്രിപ്ഷൻ, ഡിജിറ്റൽ സിഗ്നേച്ചർ, ചെക്ക്സം, ചൈൽഡ്കൗണ്ട്, ടൈംസ്റ്റാമ്പ് തുടങ്ങിയ ഘടകങ്ങൾ ഉൾകൊള്ളുന്ന ഈ സെക്യൂരിറ്റി മോഡലിൽ ആപ്ലിക്കേഷനുകൾക്ക് അനുയോജ്യമായ സുരക്ഷാ രീതികൾ അവലംബിക്കാവുന്നതാണ്. നിർദിഷ്ട ത്രീ ലെയർ ആർക്കിടെക്ചർ LXML സന്ദേശത്തെ എല്ലാ തരത്തിലുമുള്ള റീറൈറ്റിംഗ് ആക്രമണങ്ങളിൽ നിന്നും സംരക്ഷിക്കുന്നതായി കണ്ടെത്തി.

മേൽപ്രസ്താവിച്ച ത്രീ  ലയർ ട്രാൻസാക്ഷൻ മോഡൽ പരമ്പരാഗത മോഡലുകളെ അപേക്ഷിച്ചു പ്രസരണശേഷിയിലും കാര്യക്ഷമതയിലും, ഡാറ്റ സുരക്ഷയിലും  വളരെയേറെ മുന്നിട്ടുനിൽകുന്നു. ഈ പരിണാമം വിവിധ ഡൊമൈനുകളിൽ ഡാറ്റ കൈമാറ്റങ്ങളെ പരിപോഷിപ്പിക്കുന്നതോടപ്പം മികച്ച സാധ്യതകളും തുറന്നിടുന്നു.

കീവേഡുകൾ: ഡാറ്റ ഇൻറ്റർചേഞ്ച് ഫോർമാറ്റ്, LXML, ട്രാൻസാക്ഷൻ മോഡൽ, കൺസ്ട്രയിൻറ്റ് മൊബൈൽ നെറ്റ്‌വർക്ക്, സെക്യൂരിറ്റി, റീറൈറ്റിങ് അറ്റാക്സ്.

# Contents

# List of Tables

# List of Figures

# Nomenclature

**Abbreviations**

| | |
|---|---|
| ACK | Acknowledgement |
| AES | Advanced Encryption Standard |
| API | Application Programming Interface |
| ARM | Advanced RISC Machine |
| B2B | Business-to-business |
| CRM | Customer Relationship Management |
| DAO | Data Access Objects |
| DO | Data Objects |
| ERP | Enterprise Resource Planning |
| GPRS | General Packet Radio Services |
| GSM | Global System for Mobile communication |
| HTTP | Hypertext Transfer Protocol |
| JSON | JavaScript Object Notation |
| LXML | Lightweight XML |
| MEC | Mobile Edge Computing |
| NAK | Negative Acknowledgement |
| OGSI | Open Grid Services Infrastructure |
| P2P | Peer to Peer |
| PDA | Personal Digital Assistant |
| RCN | Resource Constrained Networks |
| REST | Representational State Transfer |
| RPC | Remote Procedure Call |

| | |
|---|---|
| SGML | Standard Generalized Markup Language |
| SOAP | Simple Object Access Protocol |
| WSDL | Web Services Description Language |
| WSN | Wireless Sensor Network |
| XML | Extensible Markup Language |
| YAML | YAML Ain't Markup Language |

Chapter 1

# Introduction

The increase in the use of small and portable devices coupled with wireless networks paved the way for a new trend in computing called mobile computing or nomadic computing, where users can access data at any time and from anywhere. Wireless mobile computing has established its applications in various sectors including commerce & business, telecommunications, emergency & disaster management, and real time systems. The wide acceptance of wireless mobile communication is due to the flexibility and freedom that they offer to the mobile workforce.

In a distributed wireless environment, the data will be shared and accessed by multiple mobile users. Due to this, the maintenance of consistency and integrity of shared data is much more difficult and complicated because of the inherent limitations of the wireless computing environment and the restrictions imposed due to mobility [1].

The popularity and widespread use of wireless mobile devices and wireless mobile networks coupled with the easy affordability and accessibility to the Internet has revolutionised the way people do their day to day business transactions and needs. Mobile phones are being packed with several smart features which make them capable of even processing the business logic that were hitherto done by PCs or laptops. Inherent characteristics of wireless mobile devices and networks such as mobility, constraints in resources such as energy, paved ways to new challenges to be addressed by the developers and research fraternity. A brisk walkthrough of the multifaceted features and complexities involved is outlined in the following sections.

## 1.1 Mobile Computing

Mobile computing has revolutionised the way devices communicate by providing mobility, greater convenience and enhanced connectivity. It is a technology that allows interaction between computing devices without using any physical media or links. The basic characteristics of such devices are portability over the network they are connected to. Components of mobile computing constitute portable computing devices (such as laptops, smart phones, tablets, and PDAs), mobile applications, wireless communication technologies, and clouds.

Mobile computing allowed versatility and flexibility in communication than traditional network communication. But the challenges in developing hardware components and software for mobile portable devices are quite different from conventional systems due to its inherent mobility [2][3]. This influenced both fixed as well as wireless mobile networks. These challenges include mobility, wireless connectivity, resource limitations, frequent disconnections and network issues, context awareness, multi mode operations, and security and privacy.

### 1.1.1 Mobility

Mobile devices are designed to access information and carry out operation virtually from anywhere having connectivity. In a wireless environment, both mobile devices and users may change their location during communication. To establish a connection, the system should be aware of the device location [4]. As the location of a device changes, keeping track of device location is really challenging.

### 1.1.2 Wireless Connectivity

Mobile computing depends on wireless technologies (such as Bluetooth and Wi-Fi), cellular networks (such as 4G and 5G), for accessing online services, and to interact with other devices.

### 1.1.3 Resource Limitations

Since mobile devices are designed to be portable, they have limited processing power, low memory capacity and battery life compared with their traditional counterparts. To overcome these limitations, increasing the capacity is not the adequate solution. It may increase the size or weight of the handheld device and adversely affect its portability. So designing energy efficient software and systems is the most adequate solution in this scenario [5][6]. In other words, the system should conserve power by reducing computation, data transmission, and communication.

### 1.1.4 Frequent Disconnections and Network Issues

Ensuring the reliability of wireless communication networks is really a challenging task. They are subjected to frequent disconnections and failures in the network due to many reasons such as bulkiness in transactions, inadequacy of network infrastructure, dependency to conventional networks, higher cost of maintenance, and technological handicaps [7][8].

### 1.1.5 Context Awareness

Certain mobile applications are context aware in the sense that they utilise contextual information such as movement, location, and environmental information to provide personalised services and customizations to the user [9].

### 1.1.6 Multi Mode Operations

Mobile devices support different ways to interact with the system and applications. The device may operate in different modes and they support offline mode of communication by storing and synchronising [10][11].

### 1.1.7 Security and Privacy Issues

Mobile computing exhibits unique security and privacy concerns mainly due to their inherent mobility and wireless network features. Since mobile devices connect and disconnect the network frequently, the chances of impersonation are higher. This leads to unauthorised data access and theft. Special care should be exercised during data transmission and management.

Mobile computing and distributed computing are two related concepts in networking. Mobile computing deals with mobility and portability of the devices during data processing and communication; distributed computing deals with use of multiplicity of devices coordinating together to solve certain tasks or achieve a specific goal. However, both these concepts are used together as most of the business applications are multi-tier applications whose constituents are scattered over different geographical locations.

## 1.2 Device and Network Characteristics

Mobile computing in this thesis refers to computation using mobile devices with limited capacities such as smart phones and tablets. Laptops and similar devices are excluded from this set considering their computational power, resources, convenience, affordability and data availability.

The central processing unit (CPU) on mobile devices is based on the ARM architecture known for power efficiency. ARM is a family of processors with reduced instruction architectures for processors. Clock frequencies of currently available smart phones range from 200 MHz to 1 GHz. Devices with higher frequency have higher speed. Memory availability with recent high-end models ranges from 64GB to 256 GB and are expandable further.

Major concerns that should be addressed while targeting smartphones and similar devices are as follows [12][13]:

(a) Compatibility: Different devices run different types of operating systems such as Android, iOS, and Symbian. The applications developed in one platform may not be compatible with another. So developers should consider specific features and guidelines for each platform separately.

(b) Screen Size and Resolution: The resolution and screen size of smartphone devices varies depending on product model. This necessitates the development of adaptive user interfaces that adjust itself with changes in size, resolution and orientations.

(c) Need of Optimization: Compared with laptops and computers, the processing speed and the storage capacity of smartphone devices are slower. Apart from these limitations, battery life of mobile devices is also a major concern. The portable battery is the main source of power in mobile devices. Though processing power and storage capacity in smartphone devices has improved over time, the progress in battery density is not up to the benchmark [14][15][16]. Due to these resource limitations, applications targeting mobile devices should optimise resource utilisation and improve the performance to provide better user experience.

From the perspective of mobile messaging and data interchange, the most important concern is battery life and energy consumption. Various operations such as processing, storage, transmissions and network activities drain battery power. Hence, battery power management is really a challenge in designing mobile transactions. Accommodating a battery with a large space is not a feasible solution due to space constraints [17].

The mobile communication networks are indeterminate in nature. Mobile connectivity and network characteristics play significant roles in ensuring quality and reliability in mobile communication. Some of the key characteristics include: (i) coverage and signal strength (coverage determines the area where network signal is available. Network services will be available only in the coverage area. Signal strength determines the quality of the signal available to the device. Weak signals reduce the speed and create hindrance for communication), (ii) bandwidth and latency (bandwidth is the maximum rate at which data is transmitted over the network. Higher bandwidth results in faster data transfer. Latency refers to the delay in transmission), (iii) quality of service (QoS refers to the ability of a network to set boundaries to prioritise the delivery of various types of data in a reliable manner), (iv) energy efficiency (mobile devices operate using the power provided by batteries. Network intensive operations drain the battery swiftly. So the network should be optimised for energy consumption), (v) scalability (scalability refers to the ability to accommodate an increasing amount of mobile device and network traffic), and (vi) security (should enforce robust measures to protect sensitive data from unauthorised access and theft).

The main networking technology utilised by mobile devices is the inherent phone network provided by the carrier operator. It can be 2G technologies (such as GSM, and GPRS), 3G technologies (such as Universal Mobile Telecommunications System, and EDGE), 4G technology or 5G technology. Global System for Mobile communications (GSM) is a standard for cellular communications developed by the European Telecommunications Standards Institute (ETSI). GSM offers data services on circuit - switched data channels with a maximum bit rate of 14.4 kbps. General Packet Radio Service (GPRS) is a packet switched technique that provides higher bit rates for data transmissions [18]. EDGE is an inexpensive attempt to improve the data rates of GSM without switching to 3G. Universal Mobile Telecommunications System (UMTS) is a third generation technology capable of providing Internet multimedia services along with voice [19]. 4G is a broadband standard that works with LTE and WiMAX technologies having higher bandwidth. 5G is an improvement to previous generations that offer higher bandwidth, lesser latency and high speed connectivity to the Internet. It offers services such as Ultra HD streaming videos, virtual reality and augmented reality media [20]. A comparative chart [20] showing the data rates, accessing and transmission technologies of various mobile networks is displayed in Table 1.1.

Mobile network is the only option for connecting with wider geographical areas. For short communications, other networking technologies such as Bluetooth, NFC, Wireless LAN or Wi-Fi can be utilised. Bluetooth is a very short range commu-

Table 1.1: Comparison of data rates for various mobile networks

| **Gen** | **Access** | **Transmission** | **Data Rate** |
|---|---|---|---|
| 2G | GSM, CDMA | Circuit switching | 9.6 - 236 Kbps |
| 3G | GPRS, EDGE | Packet switching | 56 Kbps - 384 Kbps |
| | WCDMA, UMTS | Circuit & Packet Switching | 384 Kbps - 5 Mbps |
| 4G | LTE, WiMAX | Packet Switching | 100 Mbps – 200Mbps |
| 5G | | Packet Switching | Up to 10 Gbps |

nication technology that uses a Service Discovery Protocol (SDP) to dynamically identify devices. Near Field Communication (NFC) is a short range wireless communication technology based on RFID that spans a few centimetres [21]. WLAN is an IEEE standard [22] for high data rate connectivity intended for business use.

## 1.3 Wireless Network and Communications

Wireless communications provide a tetherless connectivity that has revolutionised the way people communicate and access information. The term wireless communication refers to the transmission of data and information over a distance without using cables or wires or any other physical connections [23]. In wireless transmission, the medium of communication is air and data is transmitted in the form of electromagnetic signals.

Wireless communication has gained huge importance and become a motivating theme for mobile computing which fundamentally transformed connectivity and communication. There are two design perspectives for mobility: (i) mobile devices are added to fixed networks using access points; the fixed network acts as the infrastructure provider for the mobile devices, and (ii) ad hoc networks that consist of only mobile devices. Such a network does not have fixed infrastructures and its service availability is not always guaranteed [14][15][24][25].

Recently wireless communication has introduced many advanced transmission facilities such as multicarrier and channel adaptive transmission technologies. The need for high data rate services due to technological advancement in the field has imposed the following challenges in wireless communication [26].

(a) Scarcity of spectrum: The spectrum suited for wireless communication is becoming scarce due to the exponential growth in usage. In addition to bandwidth restrictions, the propagation loss is comparatively higher in wireless scenarios. This depletes the coverage area of transmission. To address this situation, the existing spectrum should be effectively utilised [27].

(b) Infrastructure need: Strong infrastructure support is needed to tackle the growing demand of wireless services. The wireless transmission system demands efficiency in resource utilisation and dynamic response for service demands in a heterogeneous environment. The main challenge to be addressed here is to improve the potential performance efficiency with reduced complexity and cost constraints [28].

(c) Energy efficiency: The transmission schemes employed in a wireless environment should be energy efficient to reduce the cost of transmission. Improving channel conditions along with its optimal usage can lead to energy efficiency.

(d) Effective accessing techniques: Wireless communication demands strict requirements such as high data rates, efficiency, low latency, and reliability. To support these requirements effective accessing techniques with mechanisms for buffering, caching, and short packet transmissions should be developed [29].

(e) Privacy and security: Since wireless networks employ broadcasting techniques for transmitting information, they are vulnerable to security threats. Innovative transmission mechanisms should be developed to maintain secrecy of data being transmitted [30].

To support a wide variety of applications that demand high data rate, reliability and seamless connectivity, wireless communication should provide efficient resource allocation and accessing schemes, infrastructure facilities, energy efficiency, privacy, and security measures.

## 1.4 Middleware System in Mobility

Middleware is a common concept used along with distributed and multi-tier systems and applications. It acts as an intermediary for facilitating integration and communication between systems and application components. Mobile computing includes a middleware system to resolve any compatibility issues or to extend existing features of the system.

The middleware system offers a standardized and reusable collection of features and functionalities that allows hiding the intricacies and complexities during interactions. The main advantage of using middleware is to support interoperability and resolve incompatibility issues that arise due to differences in platform, and language in distributed systems [14][15][31].

Common functionalities of middleware systems in mobile environments can be summarized as follows [32][33]:

(a) It facilitates communication among various components in a distributed environment.

(b) It supports integration of different applications and components. It provides a way to resolve incompatibility issues to support interoperability and data transformation as required by diverse applications.

(c) It supports load balancing and improves the scalability of the distributed system.

(d) It supports better data management through data persistence, caching, and support for abstraction.

(e) It is an option to enforce security and user authentication, and helps to maintain data authenticity and integrity.

## 1.5 Resource Constrained Networks (RCN)

Advancement in wireless mobile networks led to the development of technologies and standards such as wireless local area networks, wireless sensor networks, satellite based communication networks, wireless asynchronous transfer mode networks, mobile IP, Bluetooth communication, and IoT networks [34]. The wireless local area network covers and manages the communication within a building. The wireless sensor network consists of many small battery powered sensors connected and are mainly used for monitoring and automation purposes. IoT networks connect diverse devices and objects that are integrated with sensors and actuators for sharing data enabling automation, control and connectivity.

Each of these wireless mobile network categories have distinctive characteristics that are customised to the needs of the individual application and serve a specific purpose. However wireless mobile networks are often considered as resource constrained networks due to many intrinsic constraints and limitations such as device limitations, and issues related to network, coverage and infrastructure.

### 1.5.1 Device Limitations

Wireless mobile devices face many limitations. The screen size and resolution of a mobile device is much smaller when compared with personal computers. Such devices require a data representation with fewer character sets to suit the mobile environment. Limitation in storage capacity and caching is another constrain in the mobile environment. To improve the responsiveness of applications that depends on online / offline transactions, storage and caching is essential [2][35].

The processing capabilities of mobile devices are limited. So this limitation hinders the capability of handling and processing large volumes of data and performing complex computations in mobile devices.

Power requirements in mobile devices are usually drawn from the inbuilt battery. The capacity of the battery is comparatively less. Wireless communication, connectivity management, network and data transmission consume much power and drain the battery [36]. Similarly, wireless communication protocols such as WLAN and Bluetooth together with mobile data usage through network providers also considerably increase battery power requirements [37]. Adding multiple battery units is a solution but it increases the size and weight of the device and affects its mobility.

### 1.5.2 Network Issues

As the available frequency of the spectrum is shared between different operators, the bandwidth allotted will be limited for data transmission. This limitation leads to congestion in the network during peak hours and reduced data transmission rates. Sharing of frequency spectrum results in interference which ultimately leads to noise and degrades signal quality. These factors result in degradation in data transmission rates, and channel capacity and performance. Apart from this, the spectrum allocation policy and licensing enforced by the authorities may also lead to inefficient utilization and scarcity [38].

### 1.5.3 Coverage and Infrastructure Issues

Wireless communication demands extensive modern infrastructure support to provide connectivity. Deploying and maintaining these complex infrastructures requires large investments and resources. So, it is not economically feasible to provide connectivity in remote areas [39].

Due to these limitations, wireless mobile networks demand careful resource management, optimisation strategies, effective protocols and data interchange formats to effectively utilise the resources in hand and to offer dependable and high quality services to the mobile users.

## 1.6 Transaction Models

The term transaction model in mobile computing refers to a conceptual framework that describes the characteristics and behaviour of transactions in mobile environments. It involves execution of operations to a database or a distributed system using mobile devices. A transaction model is necessary to handle data communication and other control information transfer in a disconnected wireless environment to maintain data consistency and maximise application concurrency [40].

The constraints for managing the transactions in a mobile environment are quite different from that of conventional database oriented models. To be qualified for a mobile wireless environment, these schemes should consider the inherent characteristics of the environment such as bandwidth constraints, distributed and disconnected environment and mobility in addition to the heterogeneous nature of data. A transaction in mobile environment is quite different from traditional distributed database oriented environment [1][40][41] due the following reasons:

(a) Split transactions: Mobile transactions may prefer splitting of their computations to multiple operations that may be executed in different hosts – either mobile host or static host – depending on the situation. Such split operations may be later rejoined. Due to disconnection and mobility, there can be situations where a transaction needs to share their partial results and states to other transactions as well.

(b) Mobility: Mobility is the most important characteristic of the wireless environment. When the host device changes its location, the state of the transaction as well as the data objects changes. The stationary host should also support this feature.

(c) Mobile transactions have in-deterministic life time. Few transactions withstand for a longer time due to mobility and frequent disconnections.

(d) The chances for data replication are higher in a mobile environment. So mobile transactions should be equipped to handle frequent disconnections and failures. At the same time, measures should be taken to ensure consistency, and concurrency of shared data.

## 1.7 Open Issues in Mobile Transaction Models

The main objective in designing a transaction model is to improve the availability of data. A transaction processing model designed for the mobile computing environment should take into account the inherent limitations of the environment discussed in the earlier section. In addition to the limitations, additional challenges to be addressed while designing mobile transaction model include:

(a) Network transparency and transaction relocation: The main characteristics of mobile transactions are non deterministic lifetime and relocation. The mobile transactions follow a disconnected behaviour as the device can move from one base station to another during a transaction [42]. Due to the inability to retain such connectivity for longer time duration, a transaction is viewed as a collection of sub transactions [43].

(b) Location sensitive transaction operations: Mobile transactions should deal with location sensitive queries and such operations require manipulation of location information [44].

(c) Programming language support.

The main factors that constitute a mobile transaction model include data consistency and concurrency control, infrastructure development, performance constraints, communications costs, relocation mechanisms, user profiles and scalability [2][45]. A model that considers the above factors for dealing with data intensive transactions based on the legacy XML/HTML standard, inheriting the relevant features of existing transaction models such as splitting, low resource intensive nature, and suited for wireless environments is proposed in later chapters of this thesis.

## 1.8 Research Motivations

Advent of mobile pervasive computing facilitated a new realm of communication and computation where users can access information virtually from anywhere. This mobilization of information access led to the development of new ways of communication and user interactions. Many applications were developed that exploits the mobility feature. Existing business applications were migrated or modified to reap the harness of the technology.

A recent survey [46] revealed that 70% of the total web traffic is carried out using mobile phones. A major portion of this web traffic is generated while accessing social media and e-commerce websites. Also, people prefer mobile phones over desktop or laptop systems for accessing social media. The mobile data traffic is estimated to grow by a multiple of 3 between 2023 and 2029. Also, the average data usage per smartphone will increase by a factor of 2.7 [47].

The transition from conventional to wireless networking environment is not a gradual process and there is a scarcity of infrastructure support to backup this transition. This led to the reuse of conventional network infrastructures in wireless environments. The exponential increase in the number of mobile users and data traffic on one side, and the lack of adequate support infrastructures on the other hand is really a bottleneck in the growth of communication field.

Affordability of high end devices is still a concern in the data communication field. Even though the number of global smartphone users is 3.6 billion [48][49][50], only one-third (35.13%) of the world population afford a smartphone. While 81% of Americans own a smartphone, the smartphone penetration ratio in developing countries is still far behind (in Bangladesh, it is 5.4%, for instance). The statistics reveal that only 4% of the adult population in Ethiopia and Uganda own smartphones [48].

It is undoubtedly established that mobile devices are becoming smarter in terms of factors such as its computing power, memory capacity, battery power and veracity of applications. It does not mean that the advantage achieved in battery power is entirely available for the processing and transmission of data among the devices. Operating systems consume the biggest chunk of battery power in mobile phones. The hardware subsystems in the device consume their share of battery power (for instance, multimedia, modem operations, memory, and LCD subsystems consume 39.5%, 21.5%, 19.4%, and 17.6%, of the entire battery power, respectively). The sensor network available in the smartphone also consumes considerable share of battery power [51][52][53][54]. Power consumed by the applications and message exchange between the devices are in addition to this. The power consumed for a message exchange is found to be in proportion with the size of the message being transferred [55]. Due to the use of smart mobile phones for a variety of applications that need heavy processing, there is a scope for transaction models that can offload some of the activities from the mobile device to a middleware system [52]. The fact is that a mobile phone paves way for anywhere computing; but it is not always feasible to charge it from anywhere anytime. Hence any attempt to develop an apt transaction model that employs a lightweight message exchange format with less verbosity that can effectively conserve battery power, is a crucial step in the wireless mobile environment.

Existing data interchange formats were developed before the evolution of big data and related concepts. Hence the extent of the suitability of these formats for the current scenario is yet another bottleneck. As the requirement changes due to technological advancement, to satisfy certain requirements like managing complicated data structures, enabling streaming or real-time data processing, or integrating latest technologies, a new format might be more efficient.

Since existing formats are not tailored for resource constrained wireless environments, they face serious performance issues. Hence there is a clear scope for data interchange formats with features such as performance optimization, quick parsing, processing, transmission and data handling, that can provide enhanced application responsiveness and user satisfaction. Providing a consistent, simplified, and developer - friendly approach with such a format can make data integration easier and implementation more straightforward.

In order to enable more efficient and streamlined data interchange across different systems and applications, the central theme of this thesis is to propose a novel data interchange format that is capable of overcoming the constraints discussed earlier, enhancing compatibility, increasing efficiency, and addressing evolving data requirements. Since the targeted device is mobile, the network conditions are inconsistent, and the user and the device context is more significant, developing such a data transaction model is really challenging.

## 1.9 Research Methodology

This research work can be categorized as applied research and the main goal of this study is to improve the efficiency of data exchange in resource constrained wireless mobile networks. With the advent of mobile communication technology and availability of powerful and versatile mobile devices, more and more business applications are migrated to mobile platforms to take the advantage of data availability. But the existing infrastructure available in the wireless environment is incapable of handling this enormous amount of data generated. Since a complete revamping of existing infrastructure is nearly impossible, a better alternative is to enhance the way data is communicated.

The work on developing a new data interchange format started by analysing the performance of existing data interchange formats that are used in resource constrained wireless environments. Based on this initial study and analysis, the potential strengths and the weakness of the existing formats are identified. The study revealed that there are few formats that require improvement in messaging, while other formats need improvement in processing interfaces. The main reason for the need for improvement in messaging is mainly due to the verbosity of the underlying format. The verbosity induces many secondary issues in transmissions and storage as well. There are few formats that use complicated processing APIs that consume much time for validating and parsing the document.

The first step in the process is the development of a data interchange format. A new messaging format which is much more compact than the existing formats is developed. This format inherits the benefits of the existing prominent formats and at the same time overcomes their limitations. The format is evaluated against six performance parameters that are considered as critical in the wireless mobile environment. The values thus obtained are compared with that of existing formats to establish the advantage and relevance.

In the next phase, an architecture that makes use of this compact data interchange format is proposed. This proposed model follows a three - tier architecture based on XML/HTML based agent communication model coupled with a middleware component similar to that of Service Oriented Architecture (SOA). The client side resides in mobile devices and CRM that provides data to multiple mobile devices serves as the third tier. Since the newly proposed format is an unsupported standard for currently available CRMs, the middleware component acts as a conversion agent. The performance of this model is thoroughly evaluated and results are found to be encouraging. An important feature to accommodate transactions even in the disconnected state of a mobile device is also proposed in this phase.

As the model will be used to propagate sensitive business data in a distributed environment, chances of security breach are higher. The final phase of the work, analyses the security implications of the proposed format. It uses a three layered approach to enforce data security.

When the implementation and performance evaluation of the model is completed, it reveals that the new format has achieved its objectives and has the potential to replace existing formats in bulk business data communications.

## 1.10 Contributions

The contributions of the thesis include the design and development of

(1) Schema definition for the data interchange format proposed in the thesis.

(2) A lightweight, extensible, and platform neutral data interchange format called LXML.

(3) A three tier architecture to transform the LXML as a reliable transaction model that supports transactions in paginated, distributed, and disconnected environments.

(4) A layered architecture to check unauthorised modification of LXML messages.

## 1.11 Organization of the Thesis

This thesis is organized in eleven chapters. The first chapter introduces the topic and presents an outline about mobile computing, wireless environment and their characteristics. Chapter 2 is dedicated to discuss the state of the art in the areas of relevant technologies concerned with the thesis. In Chapter 3, the problem definition is outlined. Chapter 4 proposes the new data interchange format known as LXML (Lightweight XML). In Chapter 5, the architecture of the proposed transaction model is outlined. Chapter 6 and 7 are dedicated for the discussion on the design and implementation of the client and the middleware architecture of the proposed transaction model, respectively. In Chapter 8, the performance evaluation of the proposed transaction model is presented. Chapter 9 discusses the measures taken to ensure the protection of the LXML data from unauthorised and undetected modifications while in transit. Chapter 10 summarizes and concludes the discussion in the thesis. Chapter 11 contains the recommendations. Details about the publications out of the thesis and glossery of terms are also provided.

Chapter 2

# Literature Review

## 2.1 Introduction

This chapter makes a review of the state of the art in the following areas that are relevant to the thesis: (i) human readable data interchange formats, (ii) transaction models, (iii) middleware architecture and services, (iv) methods for disconnectedness, and (v) methods to protect data from unauthorized modifications.

## 2.2 Data Interchange Formats

Advances in the field of wireless networks and infrastructure have revolutionized the way people communicate and access information. This paved ways for the development of new applications and services that can connect and communicate with greater mobility, convenience and flexibility.

Wireless mobile devices are an inevitable component in any type of network and their presence raises a number of concerns for the current network protocols and applications. The main reason for these concerns is due to the inherent behaviour and deficiencies of wireless devices and media such as low bandwidth, high latency, indeterminate connectivity, and lack of adequate infrastructures.

Inter communication and data exchange among these heterogeneous devices in a wireless environment is really a challenging task. The major concerns while designing an interchange format are interoperability and data transfer [56]. The format should satisfy existing standards and be compatible with the existing systems, applications and operating systems; any incompatibility may lead to inefficiency.

Another point to be kept in mind while designing a data interchange format for a wireless environment is to maximize the storage and transmission efficiency [56]. To improve the transmission efficiency, the size and count of data packets should be minimal, redundant and retransmissions should be avoided, and optimal compression and encryption algorithms should be used. Lower transmission efficiency may lead to slow data transfer rates, higher congestion and latency, resulting in higher cost and decreased user satisfaction levels.

The available data interchange formats can be broadly classified as binary and non-binary formats. Since binary formats are not human readable, the thesis gives emphasis to non-binary formats. Binary formats and mechanisms such as compression - decompression are also not considered for review in this chapter due to the basic reason that the scope of the thesis is on data interchange formats that are human readable. Applications mostly rely on two standard formats for structuring and interchanging data - XML and JSON. While XML is a mark-up language based on a set of specifications, JSON is purely a data format for JavaScript applications. This section surveys the prominent non binary data interchange formats available in literature.

### 2.2.1 XML

Extensible mark-up language (XML) is the popular and the most widely used markup language due to its powerful capabilities. XML provides a standard way for representing structured and semi structured data on the Internet [57]. Structured data has a consistent format for their content and provides an insight to the role and usage of the content in the context of the document. XML is formally developed as a meta-language for creating markup languages that defines standard file format for data exchange through World Wide Web and Internet. XML is a widely accepted standard for information interchange due to its simplicity, ease of use, extensibility and self describing features [58].

XML is derived from SGML and is used for storing and transmitting data on the web. SGML is an international standard for describing the structure and content of various types of documents designed to be transmitted electronically. It is mainly used for defining markups and every XML document should conform to the SGML as well [57]. The main design goals of XML are to assist information systems in document encoding, data serialization, and Internet based structured data exchange. So XML comes with a combination of other standard family of technologies such as XSLT, XPath and XML Schema that provides a clear distinction between the document content definition and its formatting. This makes the document easy to reuse the content wherever required. The XSLT (XML Stylesheet Language Transformations) is used for applying style to the XML document. XML Path Language (XPath) is used for locating tags and XML nodes in the document. XML Schema defines the structure and rules for checking validity and well formedness of the document [59][60][61].

XML provides a standard format for data interchange and can be used across heterogeneous platforms and programming languages. The major characteristics of XML are [62][63]: (i) structured (XML allows for structured data, which means that it can be easily organized and searched. There are no predefined tags in XML and the users are free to create their own tags to meet their specific needs depending on different types of data. Its structure supports hierarchical representation of data and its tags can be nested to any level of complexity. The rule and constraints related to the structure of an XML document is defined inside the Document Type Definition (DTD). DTD is helpful in validating the document), (ii) platform independent (XML is a platform - independent format, which means that it can be used on any operating system, applications or hardware platform), (iii) human - readable (XML is human - readable, which makes it easy to understand and edit. XML documents are more content oriented and it clearly separates XML content from its presentation. This makes it a convenient format for data exchange. For presentation, it uses additional tools such as XSLT), (iv) interoperability (XML allows interoperability, which means that it can be used by different software applications), and (v) customization (XML is extensible and based on open standards, which means that it can be customized to suit specific needs. Since XML is a markup language, it provides the capability to create more powerful languages or extend capabilities of existing languages. The main reason for this feature is due to the flexibility in defining XML tags).

In addition to the above mentioned characteristics, the most promising advantages of XML include: It is based on international standards and is accepted by a wide variety of systems and applications. It simplifies the data exchange between devices and databases, improves data availability in heterogeneous systems, supports Unicode character representations, and it has a self-documenting structure with strict syntax and schemas [64][65][66].

**XML Schema**

An XML Schema is a model that defines document structure and provides a blueprint for building its contents in accordance with its structure, along with security provided by XML encryption. XML Schema defines the set of elements, attributes and data types that can be used in an XML document. It is a XML based language that simplifies the creation of sophisticated content models and focuses on reusing elements, attributes, and data types to hold important data.

The strength of XML Schema lies in the variety of data types it supports and simplifies the content description of the document. It is helpful in validating the correctness of the document in terms of its structure and content. It can be used to define allowable data patterns and restrictions, if any, in the document.

XML Schema is extensible in the sense that it can be reused or nested inside other schemas. It can also be used to define the rules for multiple documents. It supports creation of custom data types and elements while dealing with specialized data. This really saves time and effort [67].

XML Schema is vulnerable to errors and can cause serious consequences in data storage and transmission. There is no single standard mechanism to create XML Schema. XML Schema can be complex in certain situations and maintaining such a complex schema consumes much time and effort. Validating an XML document is necessary to ensure the correctness of the document. The validation process is resource intensive and hence it weakens the response time. This will adversely affect the overall system performance [68].

### Web Services and XML Protocols

Web service refers to a standardized method that allows different web based applications to communicate with each other over the Internet. It uses XML (the data interchange format), SOAP (the protocol used for data transfer), WSDL (for describing the services available), and UDDI (to list the available services) [69].

XML is the de facto standard used in web services for encoding and exchanging data between applications. It provides the flexibility and versatility to define custom tags and structures for representing data. When the client applications request for a specific data, the service responds by returning an XML encoded message that includes the required data. This XML message can be parsed to extract the data contained.

XML is often used in conjunction with a protocol. The first attempt to use XML for data interchange is with XML RPC. XML RPC is a protocol that allows invoking a remote procedure using XML messages. It is a simple, platform independent and easy to implement web service standard used in any applications. It lacks the extensibility feature that is required in distributed applications. This led to the development of Simple Object Access Protocol (SOAP) web services. SOAP is a standard protocol for sending and receiving XML - based communications over the Internet. It relies on HTTP or other related transport protocols for transmitting SOAP messages [70].

Dependency to XML - based message format and the use of advanced features like WS - Security and WS - Addressing can add complexity to SOAP web services. Processing of large SOAP messages slows the processing especially in high traffic conditions. Chances of compatibility issues and security vulnerabilities are also higher [71][72].

### XML Parsing

XML parsing is the process of reading an XML document and extracting the data contained in the document. Before parsing the document, the parsers check the well formedness and validity of the document against the XML Schema. Numerous XML parsers are available in the literature for creating and extracting XML documents. They can be broadly classified into two categories, namely, Document Object Model (DOM) parsers and Simple API for XML (SAX) parsers [73]. DOM is a tree based parsing approach where the XML document elements are modelled as the nodes of the tree. The tree nodes are traversed to read, search or update the data contained in the element. SAX parsers support an event based parsing approach where the document is parsed sequentially and generates events when it encounters an XML element or nodes. Applications must register for events to get the notifications. SAX parsers do not produce any in - memory representations as DOM parsers and are very suitable for large XML documents that cannot be kept in memory as a whole [74].

The criteria to choose a parser API for parsing an XML document depends on many factors such as user requirements, the size of the document, programming language and platform used. Parsing efficiency or performance while dealing with large documents is also an important factor to be considered.

Parsing is a resource intensive operation as it consumes a significant amount of memory and processing time to extract data contained in the document. Additionally, parsing may cause security vulnerabilities such as rewriting or injecting attacks. This underlines the need for a validation check before parsing [75][76].

### Issues of XML in Resource Constrained Environments

Though XML is a popular format for storing and transmitting structured data in heterogeneous networks, it poses the following issues especially in resource constrained networks [64][65][66][77][78][75]:

(a) Verbosity: The size of XML documents is relatively large when compared with other formats. This verbose nature of XML is due to the fact that it uses large number of descriptive tags to maintain the readability of the document [77]. In an XML document, the data is kept between tags. The users are free to choose the tags names and since there is no restriction for the length of the user defined tag name, it adds to the document size. Usage of nested tags makes the document lengthier. XML requires every element to have opening and closing tags even if there is no data content available between them.

XML supports complex data structures and data hierarchies. A document with complex data structures leads to larger size with many levels of nesting and makes it difficult to read and understand. The verbosity of XML also makes it more readable and predictable, which can be beneficial in certain contexts but is problematic in resource constrained networks where available bandwidth and memory are limited.

(b) Parsing overhead: XML parsing can take longer time due to many reasons such as verbose nature of the document, need for validation, use of nested tags and structures, use of complex data structures, and encoding and parsing methods adopted.

The main reason for this inefficiency in parsing is due to the verbosity and complex syntax of XML documents. When the size of the document increases, the time required for validating the document with the DTD increases and requires more memory to process the document. The time required for transmitting the document will also increase in proportion to the size of the document. Thus XML parsing is computationally expensive in resource constrained environments with limited processing capabilities.

(c) Need for validation: XML documents contain a document type definition that defines the document structure and specifies the rules that every document should confirm to. To ensure data integrity, every XML document should be validated against its schema before processing. This process is complex and resource intensive.

(d) Size and complexity of data structure: Appropriate data structures are designed for handling XML documents in various applications. Different approaches such as event driven approach and tree based DOM approach are available for choosing the data structure and it depends on various factors such as document size, its structure, complexity, and performance constraints. Designing data structures for handling XML is quite hard as it encourages non - relational data structures. It induces a large overhead in accessing the information contained in the document.

(e) Lack of direct mapping of object to data models: Unlike JSON and other formats, XML does not map its object directly to appropriate data models. Recently, there are some third party frameworks such as JABX developed for XML binding in Java language. Such frameworks are inherently complex and resource intensive. JABX faces performance issues while handling large XML documents. It does not support XML Schema with complex types. Maintaining such a framework is also a challenging task.

(f) Network related overheads: These are the additional overheads required for transmitting and receiving the XML documents over the Internet. These overheads depend directly on the size of the document, protocols used for transmission, encryption and decryption methods, and network characteristics such as bandwidth.

(g) Security: XML is vulnerable to many security issues. Common security issues faced by XML documents during its storage and transmission include - external entity injections and rewriting attacks. These attacks deliberately insert malicious contents to the document in order to corrupt it. These attacks are serious enough to corrupt XPath queries and result in loss of data integrity, denial of service and unauthorized content access. Rewriting attacks corrupts the code by inserting or removing malicious contents without affecting the document signature. Transmitting and processing a malformed XML corrupts the entire system and its effects will be devastating.

### 2.2.2 JSON

JavaScript Object Notation (JSON) is a light weight, text based data interchange format used to transmit data in web based applications. JSON is not a document format or a markup language such as XML; it is simply a schemaless representation of structured data [79]. In JSON, data is presented in the form of a key - value pair.

Though JSON is developed for JavaScript programming and is considered as its native format, currently it is widely supported in many programming languages, databases and web services for data interchange. It includes a minimal set of data types and is more generic in nature. JSON format considerably depends on the concepts of arrays and lists available with JavaScript. As it is a language independent format, it is a widely accepted option for porting data between applications developed using different programming languages [65].

JSON is an object serialization mechanism widely used in AJAX frameworks. The popularity of JSON is increasing due to its inherent features such as simplicity, flexibility, wide programming language and web framework support. The characteristics of JSON as an alternative to XML [80] are:

(a) Simplicity: It uses a simple and easy to use syntax for data representation. It represents data in the form of key - value pairs. It supports minimal data types and relies on arrays and lists to represent data.

(b) Lightweight: It is a less verbose data interchange format. It has a higher content density than XML. It is comparatively easy to generate and parse JSON documents.

(c) Human readable: A simple JSON document is easy to read and use by human and machine. The readability of the JSON document depends on the complexity of its structure. A document having nested JSON objects is seldom readable.

(d) Language independent: It has a consistent and well defined standard and is supported by different languages and platforms. This language neutral approach makes it suitable for integrating applications and services.

(e) Ease of parsing: Since it is a lightweight data interchange format, parsing is comparatively easier.

**Issues of JSON in Resource Constrained Environments**

Though JSON has gained popularity in recent times, it is not always the most preferred data interchange format due to the following reasons [60][81][82][83].

(a) Generic format with few data types: JSON is a generic format as it supports a wide variety of programming languages and frameworks. In this format data is represented as key - value pairs, in which key is string type and value can be string, integer, Boolean or JSON object. JSON relies much on JavaScript list and array data types. JSON supports only a few data types and does not have types to represent date and timestamp values. It is very difficult to represent complex data types in JSON.

(b) Absence of schema: JSON does not support schema and hence there is no standard way to create the document structure and rules for defining tags. In other words, there is no way to ensure the document is valid and according to the acceptable form. A tree based structure is available in JSON as a representative schema. This structure is not as capable as XML Schema. It only provides basic details such as grammar specification and preliminary semantic details to check correctness and document validity [84]. This results in interoperability issues where there are higher chances that each user may interpret it in different ways. Absence of schema may also increase the chances of security breaches.

(c) Lack of namespaces: The namespace concept available in XML is used to avoid naming conflicts among its elements. JSON does not support namespaces and uses prefixes to avoid name conflicts. Use of namespace makes the element more self describing, contextual and becomes easier to identify.

(d) Not extensible: When compared with XML, JSON format is not extensible due to the following reasons: it does not support a formal schema and relies on list and array data structures. There is no standard way to validate the document. It does not support the namespace concept as well. It uses a fixed data model with key - value pair to represent data, which is not flexible enough to represent complex data types.

(e) Verbosity: Though JSON is a compact format compared to XML, it is not the most compact and efficient format for representing data in resource constrained environments.

(f) Poor readability: Readability of JSON depends on its document structure. A JSON document with deeply nested structure will be difficult to read and extract data. Such deeply nested complex data structures incur additional processing overhead to extract relevant data.

(g) Parsing and performance issues: Even though JSON is a lightweight data interchange format, it suffers serious performance issues in certain scenarios. Parsing a JSON document with large size, complex data structures and deep nesting consumes considerable processing time and memory. Additionally, parsing performance depends on the characteristics such as the parser used, programming language, and level of validation required. The JSON requires frequent serialization - deserialization of its objects; the overall performance is considerably affected due to this.

(h) Security issues: Similar to XML, JSON is vulnerable to security breaches. JSON documents are often subjected to malicious injection / rewriting attacks during transmissions which leads to attacks such as unauthorized disclosure of information, denial of service, and data corruptions. Since JSON does not support schema, it is very difficult to validate the data at the receiving end. Sniffing JSON strings can easily identify the object properties and alter its values. So, additional care should be exercised to ensure the document validity, access control, encoding and input sanitization.

### 2.2.3 YAML

YAML Ain't Markup Language (YAML) is a human - friendly, flexible, platform neutral, and Unicode based serialization language used to store and transmit data. The applications of YAML range from configuration files to data transmission. YAML presents data in both text format and by using native data structures. It supports a variety of data types such as numbers, string, boolean, and null. Unlike in JSON which uses braces for nesting, YAML uses indentation for representing nested structures [85]. XML and YAML are not at all related to each other. XML focuses on structuring the document and imposes many constraints in that sense. But YAML never imposes structural constraints to its objects. Since there are many similarities between YAML and JSON as a data interchange format, YAML can be considered as the superset of JSON [86][85]. While JSON focuses on simplicity, universality and the ease of processing, YAML focuses on data serialization using some native arbitrary data structures.

The advantages of YAML include good readability, easiness to use, implement and release, extensible, strong expressive capabilities, good interaction between applications and programming languages, availability of native data structures that matches with agile programming languages, availability of consistent models to support generic tools, and schema awareness [85][87].

Every programming language has different levels of comfort in using YAML. It is not a preferred format for data models that need manual maintenance. Though YAML is a lightweight, schema aware serialization format, it is not as popular as JSON and XML due to the following reasons [88].

(a) Whitespace sensitivity: It uses indentation as delimiters for blocks and nested structures. Due to this dependency on indentation, it is very difficult to represent complex data structures in YAML. A minor change in indentation may result in issues in processing the elements.

(b) Limited data type support: It does not support all data types available in other serialization formats. It lacks its own definition of data types; instead it depends on data models to realize languages. As result, if compatibility is not guaranteed, portability issues may arise during data exchange. It does not support annotation of information and the use of block comments in the document. Also it does not preserve order for key - value pairs in map types.

(c) Scalability issues: Unlike other serialization formats, YAML has a different view to represent data and their relationships. Because of the syntax and structure of YAML, the parsing is inefficient and consumes much time for large dataset. Since YAML is not designed for efficiency in storage or parsing, handling complex data structures in YAML is a bit challenging and risky.

(d) Security: It is mainly used in configuration management. If not properly handled, YAML files are subjected to security risk. Additional policy files are required for imposing security measures in YAML.

The advantages and disadvantages of data interchange formats discussed so far are summarized in Table 2.1.

Table 2.1: Summary of the performance of various human readable data interchange formats

| **Format** | **Advantages** | **Disadvantages** |
|---|---|---|
| XML | Human readable, platform independent, schema aware, extensible, self descriptive using user defined tags and structures, highly popular. | Verbose with redundant tags, complex syntax and structures, higher parsing overhead, lack of support to data objects, vulnerable to security risks. |

| | | |
|---|---|---|
| JSON | Compact than XML, text based, human readable, easy to parse, language neutral and widely supported. | Lack of schema support, limited data type support, lack of support for namespace, comments, and attributes, not suitable for binary data, performance issues, vulnerable to security risks. |
| YAML | Human readable, compact syntax, uses indentation to denote structure, supports language independent object types, and use of comments, self - descriptive and expressive. | Whitespace-sensitive, no built-in schema, limited data types, prone to syntax and validation errors, security risks, not suited for bulk data. |

## 2.3 Transaction Models

Due to the wide popularity of mobile services and applications, many services have been migrated to mobile wireless platform. This necessitates the need for a mobile transaction model to maintain consistency and maximizing concurrency of mobile applications and services.

Transaction models employed in mobile wireless environment are considered as the extensions of three base transaction models [89][90][91]: (i) open nested transactions (this model is designed for long duration transactions), (ii) split transactions (such transactions can be divided into smaller, and independent transactions that are serializable and can be committed or aborted independently. The split transactions can be later rejoined), and (iii) saga compensating transactions (these are a set of independent transactions. Here each transaction has a compensating transaction that can undo the effects of corresponding component).

Various mobile transaction models such as kangaroo transaction model [92], clustering model [93], isolation only model [94], multi database transaction model [95], pro-motion model [96], toggle transaction model [92], XML / HTML based agent communication model [97][98] are reported in the literature. A discussion on the advantages and disadvantages of these models are presented below.

Kangaroo transaction model tracks data movement and behavior in a mobile host transitioning between cells in a static network. It employs global and split transactions using a data access agent at the base station. The agent processes user transactions and commits them to the server. For each query, the agent creates two transactions—global and local, known as joey transactions, to manage the base station's scope. When devices switch base stations, control transfers to a new agent. The merits of the model include: (i) transaction splitting, (ii) sequence numbering for each transactions, and (iii) support for status monitoring and tracking [92].

In this model, the communication is always channelized to the databases either in the base station or in the host device. It never makes use of a consistent format for messages transmission. This model induces additional overhead to maintain the data access agent [92].

The clustering model is an open nested transaction model for fully distributed systems, organizing data into clusters based on their meaning or storage location. These clusters, which can be static or dynamic, contain mutually consistent data. The consistency level varies with network bandwidth accessibility between clusters [93].

The merits of this model include: (i) suitable for distributed systems, (ii) support for both connected and disconnected modes for transaction execution, and (iii) ensuring data consistency within a fully distributed environment [44]. The demerits of the model include: (i) a database oriented model with higher maintenance cost, (ii) scalability issues and difficulty to manage large number of clusters, and (iii) overhead for communication within cluster and outside [92][89].

Isolation only model is designed for the Coda file system. The model extends regular file operations. Coda, a distributed file system, uses file hoarding and concurrency control to support mobile clients in disconnected mode, resolving read and write conflicts based on operation significance [92][99]. The merits of this model are its ability to resolve read/write conflicts effectively and its support for database communication in both disconnected and distributed environments. A drawback of this system is its compatibility limitation to devices not supporting coda file systems [92][94].

This model uses a replication scheme with one master copy and many replicas, handling base transactions on the master and tentative transactions on replicas. During disconnection, tentative transactions work on replicas and convert to base transactions upon reconnection. While supporting disconnected distributed transactions, the model faces limitations like processing overhead from multiple transaction executions, managing multiple data copies, and ensuring data integrity and consistency with replicas [92][94].

Multi database transaction model is designed for a multi-database environment. The model handles messages from a mobile host to its coordinating site asynchronously, allowing disconnection without interruption. Each workstation has input, output, message, and transaction queues to manage local or sub-transactions. The model's highlight is its queuing and concurrency control mechanism. However, it relies on conventional databases for mobile communication and the numerous queues can cause bottlenecks [44][92][93][94][95].

Pro-motion model is based on the nested transaction model discussed earlier [96][100]. It facilitates a distributed and disconnected mode of transaction processing using client - server architecture. Mobile transactions are organized as nested transactions, where the top - level transaction is executed on fixed hosts, and sub - transactions are carried out on mobile hosts [101]. Disconnected transaction processing is a prominent feature in this model [101]. The benefits of this model include: (i) support for nested and sub - transactions, and (ii) disconnected architecture [92]. However, the main drawback of this model is its high resource requirements on the mobile host [92].

In the toggle transaction model, a mobile multi database system is formed by assembling a set of mobile databases. A multi database management system is developed to operate on these set of databases. This model employs a support station to handle global transactions. When the device changes its location, another support station handles transactions [92].

XML / HTML based agent communication model is a distributed processing architecture based on XML/HTML is discussed in [97] and [98]. In this model, XML is used as a payload for data exchange among nodes. The XML/HTML based Agent Communication Model [97] operates on a mobile agent platform utilizing Agent Communication Language (ACL) messages for both inter - agent communication and inter - platform migration. Its key advantage lies in offering distributed processing capabilities very similar to client - server systems. Data transmission to client nodes occurs via XML payload. However, the model is marked by certain drawbacks, outlined in [97]: (i) a fixed / fully connected network architecture architecture necessitating agent module installation on communicating base stations, (ii) the absence of a message queuing mechanism, and (iii) the lack of support for split transactions.

The concept presented XML/HTML based approaches can be extended to the resource constrained mobile environment due to following reasons [97][98]: (i) ability to encode different types of messages in a simple and convenient manner at the same time it is easy to change, (ii) simplifies the architecture of agent systems while building programmable information base in XML, (iii) as agents can directly read and manipulate data contained in XML, it is easier to enforce access control policies, and (iv) less performance overhead when compared with other traditional database oriented models.

The performance of various mobile transaction models available in the literature are summarized in Table 2.2.

Table 2.2: Summary of the performance of various transaction models

| **Transaction Model** | **Advantages** | **Disadvantages** |
|---|---|---|

| | | |
|---|---|---|
| Kangaroo transaction model | Sequence numbering, transaction splitting and status monitoring | Supports base station to device communication only in fixed networks, no specific message format |
| Clustering model | A model for distributed system with connected and disconnected mode of operation | Scalability issues, higher maintanance cost and processing overhead |
| Isolation only model | Database model that supports connected and disconnected modes, handles read write conflicts | Compatible with coda file system only |
| Multi database transaction Model | Queuing and concurrency control mechanism | Not suited for mobile environment due to the dependency with traditional database system, congestion due to queueing |
| Pro-motion model | Distributed and disconnected mode of transaction processing using client-server architecture | High resource requirements |
| Toggle transaction model | Split transaction, multi database model | Fixed network, complex |
| XML / HTML based agent communication model | Distributed, client - server based message exchange using XML | Fixed or fully connected network architecture, No queuing mechanism, No message split / rejoin |

From the literature, it can be seen that a lot of issues are to be addressed in this area such as [89][90][92]: (i) all the available transaction models are database oriented, (ii) only few of them support (such as clustering model, two - tier transaction model, pro-motion transaction model, and multi database transaction model) support distributed and disconnected architecture in the database level, (iii) since the transaction models are database oriented, predicting the behaviour of these models in mobile data connection is challenging, (iv) queuing mechanism is applied only in multi database transaction model – but the transaction queuing creates bottleneck, (v) proper acknowledgement mechanisms are not addressed, (v) particularly in XML based message exchange models, the huge verbosity of XML format can be a limiting factor for the resource constrained devices and wireless networks, (vi) data synching mechanism is not adopted, and (vii) lack of a middleware component to support heterogeneous server communication.

Mobile wireless environments have to deal with heterogeneous types of transactions and message exchanges. As the purview of transactions and exchanges cannot be limited to database oriented activities alone, none of these models are suited for data communication in mobile wireless environments. It can be concluded that the XML/HTML based model is the model that can be considered as the base model for the proposed work. This model is expected to use a lightweight data interchange format as its payload.

## 2.4 The Middleware Components

In a distributed environment, middleware is a software layer that homogenizes infrastructure diversity. Its main goal is to provide services and abstractions to eliminate heterogeneity, defining interaction paradigms for coordination. Middleware simplifies distributed application development by managing data sharing, communication, security, and other concerns, enabling seamless interoperability across systems. It is crucial for addressing the growing complexity of distributed systems.

In this section, the state of the art in various middleware systems and their features are explored.

Service Oriented Architecture (SOA) is a software development method using reusable components known as services. Services process data or perform specific tasks and return responses. SOA, a popular design for distributed systems, supports modularity, reusability, and scalability. Middleware in SOA handles business logic, offering service encapsulation and flexibility, and resolves technological heterogeneity. It also manages distributed security, performance, and scalability. This section discusses the state of the art in middleware architecture and SOA [102].

SOA is the ideal and most widely accepted paradigm for mobile oriented services due to its agility, flexibility, dynamic and loosely coupled platform neutral nature [103][104]. It is widely used along with SOAP, WSDL and ESB implementations. The services are offered using standard protocols including SOAP / HTTP or Restful HTTP to send requests to read and / or update data. These services are independent, self - contained and provide agility and improved interoperability in business systems. SOA offers the following merits over the conventional enterprise application development [105][106][107]: (i) agility and flexibility: SOA allows developers to assemble applications from reusable components, accelerating the software design process and enabling quick responses to new business opportunities, (ii) extensible: The functionalities developed for a particular environment or platform can be seamlessly extended to a new environment without much effort, and (iii) collaboration: SOA enables improved collaboration with business and technology. This enables us to define the business scope in terms of services.

In spite of the fact that the SOA architecture is a natural option for business application development, it has few disadvantages [105][108][109]: expensive (implementing SOA is costly due to infrastructure, service interfaces, governance, tools, and training. High expertise is also required to manage SOA), high response time (SOA's full validation of input parameters and additional communication layers introduce performance overheads, resulting in high response times), high resource consumption and bandwidth (this is due to verbose XML messages. Poorly designed service interactions can lead to excessive communication and resource use, making performance optimization challenging in resource-constrained environments), integrating existing systems with new services is complex and time-consuming, and performance optimization in heterogeneous distributed environments is difficult.

Representational State Transfer (REST) is an architectural style for designing networked applications. It offers a simple and scalable alternative to SOA for web services and APIs [104]. REST provides a lightweight, flexible, secure, and uniform interface for interconnecting web resources, mainly used in client-server systems, allowing independent evolution of client and server components [104][110].

An integrated architecture for connecting mobile systems to the SAP enterprise server is proposed in [111]. It uses a lightweight OSGi-based platform [111]. It facilitates communication between mobile devices and the SAP backend without personal computers, handling connectivity, data management, compression, and transfer. Data is transmitted as compressed XML business objects (BO) between the client and SAP server, with RFC calls used on the server side for data transmission. This platform possesses the following demerits [111]: (i) since the framework is dependent on XML for communication, it is prone to all inherent drawbacks of XML including verbosity and huge processing requirements, (ii) even though the generated BO is much smaller than the original BO, the total size of instance might be still too large to be accommodated in the mobile memory, (iii) it uses B Trees or B+ Trees for implementing the model and uses heavy indexing, making it more complex to handle, and (iv) the framework incurs additional overhead to compress the XML to make it lightweight.

MobileSOA applies SOA principles to lightweight mobile environments, based on Web 2.0 [104][112]. It virtualizes local, remote, and ambient services uniformly, accessed via a Web 2.0 front end. Mobile devices act as both producers and consumers of web services using REST asynchronous interfaces. Challenges include managing service dependencies, interoperability with conventional SOA applications, limited bandwidth, connectivity issues, and device performance optimization. Consequently, many enterprise system features are not addressed by this approach [110].

A data delivery framework for Mobile P2P communications [113] uses a reputation value from neighboring peers for data dissemination in wireless ad hoc environments. It employs proprietary schemes and mobile scheduling techniques, using location-based methods like Euclidean distance and rectangular partitioning for high data deliverability. However, it overlooks energy consumption, power levels, and the cost and performance overhead of distance calculations. Communication between host servers in heterogeneous environments is not addressed. Other P2P data dissemination models include pure flooding, where messages are forwarded by peers until they reach the target, and trustworthiness-based broadcast (TBB), which prevents redundant transmissions using multipoint relaying [113][114][115][116].

Mobile edge computing (MEC) offloading brings computation and data storage closer to where it is needed, at the edge of the mobile network. This distributed computing architecture extends cloud resources to the network edge, reducing latency, improving response times, and enhancing network resource efficiency. However, MEC can cause network congestion, incur costs, and face scalability issues. It also struggles with task partitioning, transparency, portability, and is vulnerable to physical and security threats [117][118][119].

Performances of various middleware components are summarized in Table 2.3.

Table 2.3: Performance summary of various middleware components

| **Architectures** | **Advantages** | **Disadvantages** |
|---|---|---|
| SOA | Agility and flexibility, extensible, collaboration, suited for multi - backend systems and integration | High response time and resource consumption due to the XML dependency, performance optimization required |
| REST | Simple, flexible, stateless and compatible with existing technologies | Lacks flexible data queuing mechanism, not suited for multi - backend systems, not a design for integration |
| OSGi SAP Server | Lightweight architecture, better data management, compression and transfer schemes | Support for SAP server only, prone to all inherent drawbacks of XML, complex and incurs additional processing overheads |

| | | |
|---|---|---|
| MobileSOA | Flexibility, scalability, and enhanced user experiences | Complexity, performance concerns, and resource consumption |
| Data delivery framework for Mobile P2P | High data deliverability and multi point data dissemination | Does not consider energy consumption and power level, broadcasting leads to redundant transmissions of messages, causes overloading and congestions |
| MEC offloading | Reduced latency, improved response time, and more efficient use of network resources | Causes network congestion, cost and scalability issues, challenges in partitioning, providing transparency and portability of tasks, vulnerable security threats |

The key challenge in a distributed environment is to share data and collaborate between multiple sets of users without sacrificing the consistency and integrity of data. Each of the above mentioned challenges have a serious impact on performance and reliability of the system. Utmost care should be exercised in planning, designing and maintaining such systems to address these challenges. In spite of the challenges faced by SOA middleware architecture, it is the best suited model to incorporate as a middleware component in the proposed system due to the following reasons:

(i) Loose coupling: SOA focuses on loosely coupled services that allows easier integration and modification without affecting other components of the system.

(ii) Interoperability: SOA facilitates interoperability, enabling different services to interact seamlessly regardless of underlying technologies, platforms, and languages.

(iii) Scalability and flexibility: The modular structure of SOA facilitates scalability through the addition or modification of services as and when needed. It easily adapt to changing requirements.

(iv) Modularity and reuse: SOA promotes the development of reusable services. By creating services that may be shared by other components or applications, middleware systems can reduce development time and effort.

(v) Abstraction: A layer of abstraction offered by SOA enables services to conceal their underlying complexity.

## 2.5 Ensuring Disconnectedness

Disconnected mode ensures client devices can access and communicate critical data during temporary failures of the shared repository. In distributed environments with portable devices, unpredictable connectivity makes this mode essential. When devices rejoin the network, all critical messages must be transmitted to maintain data integrity.

Disconnectedness is primarily offered by cashing of data, shared file systems, message queues, and middleware systems [8][120][121].

### 2.5.1 Disconnectedness using Caching of Data and Shared File System

Caching data and shared file systems support disconnected mode operations, but mobile computing limitations make this challenging. The Coda file system addresses these issues, enhancing data availability in distributed systems through caching [8]. Each Coda client has a local disk and connects to servers via a high-bandwidth network. During server or network failures, Coda ensures ongoing data access using two strategies: disconnected operations and server replication, relying on an optimistic replica control approach where clients use cached copies of data [120][121].

In Coda, cache misses aren't always transparent, and significant misses can disrupt transactions. Longer disconnects increase resource depletion and conflict chances, requiring manual intervention [120]. The system is vulnerable to loss and destruction. While Coda adapts to low bandwidth, change propagation time and cache memory requirements are concerns in resource-constrained environments. Its complex priority-based cache management and conflict resolution add significant processing overhead. Thus, Coda isn't suited for wireless mobile units, facing scalability bottlenecks as user numbers increase and coordination becomes difficult in varied networking scenarios [121].

### 2.5.2 Disconnectedness using Message Queues

In this model, a message queue serves as an intermediate repository for data exchange. The sender transfers data to the queue, and the receiver fetches it asynchronously, detaching once communication is completed.

In long-distance communication without queues, each intermediate node must always be available for message forwarding. A queuing system stores messages at intermediary nodes until they're ready for transmission, supporting asynchronous communication. Without queuing, a business transaction system must maintain many connections, and a single failure can render the system inoperable [122][123].

There are different types of message queues [123]: (i) point to point queues (a single message is added to the queue and one application accesses it. Sender and the receiver must be known to each other), and (ii) publish - subscribe queues (a copy of message generated is broadcasted and every interested application subscribes it. This method supports decoupling and publishers and subscribers are not required to be known in advance). The main benefits of using a message queue include [123]: (i) it supports asynchronous mode of communication without direct connection and is independent of time, (ii) message prioritization, (iii) support data recovery, and (iv) higher data integrity, and security. Though, the queuing model supports message prioritization and reliable delivery of messages, shared access is limited and time consuming [124].

Message queues support seamless communication and coordination among the components of a distributed application. Message queues are the best option to provide disconnectedness in distributed environment due to the following reasons [125]: (i) optimization and better performance: queues support asynchronous communication where both the sender and receiver interact with the queue and not directly with each other. This arrangement ensures that no component has to wait for another, enhancing the optimization of data flow, (ii) improved reliability: message queues ensure data persistence and minimize errors during system downtimes. Enhanced fault tolerance can be achieved by using queues between components; even if one part becomes inaccessible, others can still interact with the queue. Mirroring the queue further improves availability, (iii) scalability: queues eliminate the risk of collision of requests from different components and facilitate effective distribution of workloads, and (iv) support for decoupling: message queues provide a streamlined way for decoupling distributed systems.

### 2.5.3 Disconnectedness using Middleware Systems

Middleware systems enable applications to function in intermittently connected environments by implementing disconnected mode operations through: (i) caching (manages local caches of frequently used data for client devices, allowing access to recent data during disconnection. However, it can cause network delays and is vulnerable to failures), (ii) buffering (holds user actions and data modifications made offline, synchronizing changes when connectivity is restored, ensuring data isn't lost), and (iii) push-pull communication (push refers to the server updations to the clients upon reconnection and pull refers to the notification by the clients to the server upon reconnection and request updates). Frequent push-pull can lead to network congestion and potential data loss [126][127][128][129].

From the literature, it is clear that message queues are the best option to provide disconnectedness in a distributed environment. A summary of the performance of various methods used to support disconnectedness is given in Table 2.4.

Table 2.4: Summary of the performance of various methods used to ensure disconnectedness

| **Methods** | **Advantages** | **Disadvantages** |
|---|---|---|
| Caching of data and shared file system | Optimistic replica control approach | Resource scarcity and conflicts |
| Message queues | Asynchronous communication, message prioritization, support for reliability, integrity, scalability, data recovery, security, fault tolerance, and decoupling of components, suited for distributed environments | Shared access is limited and time consuming |
| Middleware systems | Proper data synch, fast updation | Network delays, vulnarable to failures, congestion and suffers potential loss of data |

## 2.6 Security: Possible Attacks and Solutions

Data transmitted over the network is vulnerable to many types of attacks. Depending on binary serialization formats alone cannot safeguard the data and guarantee secure transmissions across the networks. A robust and reliable security schema is essential to maintain the integrity and confidentiality of data. Following are the major attacks on web services and SOAP messages [130][131]: (i) parameter tampering attacks: these attacks are also known as input validation or data manipulation attacks that involve changing application data, including user passwords and permissions and other information, by altering parameters that are communicated between the client and server. The XML injection and SQL injection attacks are two different types of parameter tampering attacks, (ii) rewriting attacks: a message is intentionally altered or corrupted by attackers without modifying its signature. The attackers manipulate the content of the message, attempting to deceive the recipient without being detected by the authentication mechanisms in place. Despite the message being changed, the signature on the message remains intact, making it appear genuine and trustworthy, (iii) denial of service (DoS): these attacks are carried out to disrupt specific services or the overall functioning of an entire application as intended. These attacks are typically executed by depriving the application resources, making it unavailable to users. Various methods to achieve this attack include intentional data overwriting and overloading servers with a flood of requests, leading to slowed performance, unresponsiveness, or complete down, and (iv) error handling: attackers may retrieve sensitive information about the internal state of web services by catching uncaught exceptions.

This section focuses on rewriting attacks due to the following reasons [130][132]: (a) wide impact: rewriting attacks pose a substantial security risk to system functionalities, especially considering the widespread use of XML in data exchange and storage representation among various applications, (b) data integrity and confidentiality: rewriting attacks can result in the modification of both its structure and content, leading to the corruption and inaccuracy of the information. This causes a significant threat, especially when dealing with sensitive data such as personal, financial, and government policy related information. The consequences of such alterations can be severe, potentially causing irreparable damage to the integrity and reliability of the data, and (c) fraud and impersonalization: rewriting attacks can lead to fraudulent transaction, identity theft, and impersonalization particularly in financial transactions. Moreover, such manipulations can result in non - compliance of regulations inviting legal actions, penalties and reputation damage.

Several standards such as WS Security, SOAP Account, WS Policy and WS Policy Advisor are available in the literature to check rewriting attacks. The approaches used to prevent rewriting attacks in web services can be categorized into policy - based approaches, inline approaches and string based approaches [133].

The integrity of SOAP messages is guaranteed by XML digital signatures. Before sending the message, the signer creates a hash of its contents. After receiving the message, the receiver recalculates the hash for the received content. Any variation in the hash indicates a content change. Digital signatures can be applied to any portion or even a specific element and are based on industry standard encryption methods. Implementing digital signature and the verifying process is very complex as it necessitates resource intensive operations such as computation of hash values and message decryption. The main limitation of a XML digital signature is that a signed XML element and along with its signature can be moved from one part of the document to other while maintaining the validity of the signature. This behaviour is mostly caused by the fact that the location of the signed element inside the SOAP message hierarchy is not taken into account by the signature. This limits the scope of an XML digital signature and could be exploited by the attackers to perform unauthorized changes [134][135].

Rewriting attacks in XML files can be detected and prevented using Web Service Policy (WS Policy). In this approach, appropriate security policies are recorded in a policy file. Such policies impose a position for each signed XML element in the policy file, and the person enforcing the policies should exercise utmost care while prescribing security policies. A policy advisor tool to analyze the SOAP messages is demonstrated in [135][136].

Bit Stream [137] is another policy - based approach that can locate and fix SOAP faults. This method relies on SOAP elements to auto detect defects and problems. Despite of its benefits, there are a number of issues with these techniques such as (i) as WS Policy and WS Security rely on the XML digital signature specifications, these techniques will inherit the problems associated with employing XML digital signatures, (ii) atmost care should be taken while framing policies because any loop holes while prescribing policy can cause serious detrimental effects, and (iii) the performance of policy advisors tend to degrade as the size of the policy file increases [138][139][140].

The structure of a SOAP message serves as the foundation for inline based approaches, which modify the SOAP header by inserting additional elements to it. The updated header is referred to as SOAP Account and it includes vital information such as count of header elements, counts of child elements in the body, reference counts for signing components, information about its parent, and child and sibling objects. X.509 certificates are used to sign the SOAP accounts. The inline techniques provide a novel idea to prevent rewriting attacks; it cannot guarantee to prevent all types of attacks and is not efficient due to high complexity [139][141].

RewritingHealer presented in [141][142] is an extension to the inline approach that incorporates additional information such as depth of the signed node and its parent elements.

The XML - based attacks Tolerant SOAP (XaT-SOAP) approach [143], bifurcates the SOAP message to its content and header for enforcing security. A SOAP message structure is framed first. Then a SOAP message is created according to the structure and later the header is attached to it. This approach provides a provision to validate the message and to detect rewriting attacks at the receiving end. This approach exhibits significant improvements but still has serious flaws [141][143].

The string based approach [141][144] takes into account the absolute path from the root node to the signed element using a subset of XPATH expressions referred to as FastXPath. In normal circumstances, this strategy can protect the message with minimal performance impact. But when the depth of the element increases, the size of the SOAP message rapidly increases. This causes additional processing overheads and performance issues. The cost for detecting security breaches is linearly dependent on the depth of the XML. This is the main disadvantage of string based approach [145].

To deal with Denial of Service (DoS) attacks in web environments, a self - adaptive multi - agent architecture is proposed in [146]. The proposed method uses a hierarchical classification approach with two phases. In the initial phase, a case based reasoning engine is used for filtering attacks. The method is very complex as it involves decision trees and neural networks for classification [146].

Fault Tolerant Web Services (FT-SOAP) is a fault tolerant framework proposed for protecting SOAP messages from security attacks [147]. This framework is based on the object fault tolerant service in COBRA architecture. Since COBRA architecture is entirely different from that SOAP, the validity and performance is yet to be proved.

A summary of the performance of various works to check rewriting attacks as shown in Table 2.5

Table 2.5: Summary of the performance of various techniques for resolving rewriting attacks in XML

| **Methods** | **Advantages** | **Disadvantages** |
|---|---|---|
| Web Service Policy (WS Policy) WS Security | Policy based approach, supports granular control | Complex to write policies and maintain it, incurs additional overheads, generates false positives |
| XML Digital Signatures | Standardized way to ensure data integrity and authenticity | Overhead for generating and validating signature, difficulty in key management, prone to security attacks |
| Bit stream | Policy based approach, uses SOAP elements to auto detect defects and problems | Inherits the problems associated with XML digital signatures, complexity in writing and managing policy files |
| RewritingHealer | Inline approach, takes into account the depth of signed nodes | Inherits the problems of digital signature, complex to implement |
| XML - based attacks Tolerant SOAP (XaT-SOAP) approach | Inline based approach, provides provision to validate the message, efficient | Cannot detect all types of rewriting attacks |

| | | |
|---|---|---|
| FastXPath | String based approach, protects all reasonable cases of attacks | Requires long calculation time in order to determine the structural information, processing overhead depends on document size |
| Self - adaptive multi - agent | Based on classification and successfully defends DoS attacks | Used for DoS attack only, classification is a complex process involving decision trees and neural networks |
| Fault Tolerant Web Services (FT-SOAP) | FT-SOAP is interoperable, less processing overheads | Only for COBRA, complex to implement, validity and performance is not analyzed for SOAP |

Several security standards and solutions have been developed to address rewriting attacks and enhance the overall security of data and applications. Commonly used techniques to thwart unauthorized modifications include digital signatures, data encryption, and secure communication protocols.

From the literature, it is very clear that many of the XML security standards developed till date are based on concepts that overlap each other. Therefore, the issues pertaining in one standard are carried over to the other. XML based methods inherently suffer from message verbosity, causing performance related issues. Most of these standards are prone to performance degradation and memory inefficiency while implementing [148][149].

As there is no single solution to tackle rewriting attacks, a multifaceted combination of these techniques, including input validation, encryption, secure coding practices, adherence to security protocols, and ongoing security assessments, is vital in mitigating the risks associated with rewriting attacks and maintaining the integrity and confidentiality of data. Thus, security in data transmission is always implemented in different layers to ensure the integrity and confidentiality of the data. This delegation of security measures to different layers of a communication system ensures the data is protected even though there is a breach in one layer or the other and the overall system reliability is not compromised. A layered approach is most suited for data transmissions due to the following reasons: (i) defence in depth approach (where each layer contributes to the overall system security. A security breach in one layer can be tolerated and remaining layers act as a barrier to the attackers guarante the overall system reliability), (ii) abstraction (makes it easier to develop and maintain security measures in each layer without affecting other layers), (iii) adaptation (system can add or skip certain layers to meet specific security requirements), (iv) flexibility and interoperability, and (vi) reduced complexity and maintenance cost.

## 2.7 Summary

This chapter discusses the state of the art in four key characteristics of transaction models for mobile wireless environments: data interchange formats, transaction models, middleware architectures, methods for providing disconnectedness, and methods for preventing unauthorized data modification.

Maximizing benefits in wireless mobile transmission requires modern infrastructure, support services, and specialized management. However, infrastructural growth lags behind technological advancements, forcing stakeholders to reuse conventional networks. Improving connectivity and enhancing existing systems, protocols, and standards is essential to address this issue.

Human-readable non-binary data interchange formats like XML, JSON, and YAML are verbose and unsuitable for mobile wireless environments. Their limitations highlight the need for a novel format that improves data transmission in resource-constrained settings, retaining XML's advantages while being less verbose and lightweight.

Mobile transaction models reported in the literature focus mainly on database-oriented approaches, which are inadequate for the diverse transactions and message exchanges in mobile wireless environments. Thus, these models are unsuitable for such environments. However, XML/HTML-based approaches offer viable features for resource-constrained mobile settings.

Prominent middleware architectures show that SOA-based architecture is dominant. It enables seamless communication between components as services, structuring software as a collection of reusable, loosely coupled services with well-defined interfaces. SOA middleware offers tools for service discovery, invocation, and coordination, simplifying these processes within the application. For large-scale distributed networks, designing software with openness and standardization to interact with various services is essential.

Due to unreliable and unpredictable mobile network conditions, a disconnected mode of operation is essential. A queuing model is the most suitable for this, offering asynchronous communication, reliability, fault tolerance, scalability, and component decoupling. These benefits ensure seamless, efficient communication and reliable message delivery despite network interruptions.

Unauthorized message adulteration during transit is a significant concern. Most literature focuses on protecting XML documents, but many XML security standards overlap, inheriting each other's issues. XML's verbosity leads to performance and memory inefficiencies. Thus, there is a clear need for a novel transaction model suitable for resource-constrained wireless environments.

## 2.8 Research Gaps Identified

The literature survey gives light into the following gaps:

1. Need of a schema-aware, human-readable, platform-independent, extensible, and lightweight data interchange format suitable for wireless mobile environments as an alternative to widely-used formats like XML and JSON.

2. Development of a lightweight and reliable transaction model capable of supporting transactions in distributed wireless mobile environments, including support for paginated mode and operations during network disconnections.

3. Implementation of mechanisms to prevent unauthorized modifications of messages during transit, ensuring message integrity and security in wireless mobile communication scenarios.

Chapter 3

# LXML - The Proposed Lightweight Data Interchange Format

## 3.1 Introduction

An effective data structuring and formatting mechanism is essential for managing data traffic, to prevent excessive bandwidth usage and for ensuring portability & interoperability in a resource constrained wireless environment. The data transmission rates and performance can be significantly impacted by the selection of an appropriate data interchange format [117][150]. Efficient communication between the devices helps in mitigating the infrastructure costs to a great extent.

In this chapter, a novel data interchange format called Lightweight XML (LXML) is proposed as an alternative to prominent data exchange formats such as XML and JSON. The format is lightweight, schema aware, extensible, language and platform neutral, and human readable that is well suited for resource constrained wireless environments.

## 3.2 Characteristics of a Data Interchange Formats

A data interchange format defines the ways the data is communicated between the devices and applications. In a heterogeneous environment with multiple data sources and clients, the data interchange format plays a crucial role in communication and its efficiency [88][118][119][151]. The characteristics of a data interchange format are simplicity, generality, unambiguity, extensibility, completeness, schema awareness, ease of implementation, platform neutrality, adherence to existing standards and availability [88][119][150].

(i) Simplicity: A non binary data interchange format should be human readable and simple to use and understand. As the complexity of the format increases, the overhead to transmission, parsing will be increasing dramatically.

(ii) Generality: The format should be independent of the platform, communication protocols and other methods used to exchange information. It should be applicable in all circumstances and scenarios that are pertinent to a particular domain.

(iii) Unambiguous: The format should represent the dataset in an unambiguous way. Any attempt to make the model precise, should not affect the semantics of the data exchanged. The model should act as a blueprint for easy mapping of data in any domain.

(iv) Extensibility: The format should be extensible in the sense that information can be included without any additional overhead. The format should be a free - form structure that does not impose any structural restrictions. This property is crucial due to the reasons such as system upgrades, technological enhancements and additional requirement enhancement. Hence the format should encourage the development and integration of related tools and applications.

(v) Completeness: The model should offer the ability to express and represent all pertinent characteristics of the domain. Variety of dataset or dataset standards may be used for exchange and the model should support federations for these dataset.

(vi) Schema aware: Any data interchange format used in data communication should adhere to a schema to avoid data inconsistency and transmission flaws.

Schema imposes a structure to the document and the validity and well formedness of the document can be ensured based on the schema.

## 3.3 Lightweight XML (LXML) Format

XML is a hierarchical modelling language that uses user defined tags to represent parent – child relationships. The proposed LXML format is based on XML. It uses a hierarchical representation that never uses user defined tags to hold the data as in XML. In any hierarchical representation, data items will be placed in different levels. The top most tag will usually be a container tag that holds other containers or atomic tags. This top container tag is kept in level 0 and its immediate child is considered as level 1. A document contains many levels with a mix of atomic and container tags in hierarchy. The deeper the data goes, the level number increases.

The LXML is a level number based hierarchical representation. The top most tag in the document called the root tag is kept as such to identify the transaction or the document. The immediate child nodes of the root tag are marked as level 0 and the level numbers of successive child nodes increases by 1. All sibling nodes at a particular level will share the same level number. Table 3.1 shows an XML document with level number identified for each tag.

To create a LXML message for a given set of data, the root tag is kept as such without tags symbols or any angle brackets. The data items are included by appending their corresponding level numbers as a pattern **<Level Number> data**. A sample LXML message generated for a given simple XML document without any container tags is shown in Table 3.2.

In Table 3.2, the root tag product is a container tag with three simple and atomic child tags. These entire nodes are at the same level 0. In many scenarios such as e - commerce and business transactions, the XML document contains repetitive container tags as shown in Table 3.3. In such cases, the nested container tags are considered as level 0 and its child tags constitute level 1.

In business transactions, a combination of simple atomic tags and nested con-

Table 3.1: An XML document with level numbers marked for its tags

```
<products>
     <brand>soniscorp</brand>              --> level 0
     <prodyear>2020</prodyear>             --> level 0
     <product>                             --> level 0
           <name>television</name>         --> level 1
           <date>
                 <month>Jan</month>        --> level 2
                 <year>2020</year>         --> level 2
           </date>
           <price>1000</price>             --> level 1
     </product>
     <product>
           <name>refrigarator</name>
           <date>
                 <month>Jan</month>
                 <year>2020</year>
           </date>
           <price>1500</price>
     </product>
</products>
```

tainer tags together constitute a transaction. Such atomic tags are mostly used to provide additional description about the container blocks. While creating LXML messages for such XML structures, a sudden decrease in level number will be observed to depict the level changes. The LXML message generated for an XML document with mixed tags are shown in Table 3.4.

Table 3.2: The LXML message generated for a simple XML message with no container tags

| XML Format | LXML Format |
|---|---|
| `<product>` | `product` |
| `<name>television</name>` | `<00>television` |
| `<manufacturer>sonis</manufacturer>` | `<00>sonis` |
| `<price>100</price>` | `<00>1000` |
| `</product>` | |

Table 3.3: LXML message for an XML with nested container tags

| XML Format | LXML Format |
|---|---|
| `<products>` | |
| `<product>` | `Products` |
| `<name>television</name>` | `<00>` |
| `<brand>soniscorp</brand>` | `<01>television` |
| `<price>1000</price>` | `<01>soniscorp` |
| `</product>` | `<01>1000` |
| `<product>` | `<00>` |
| `<name>refrigarator</name>` | `<01>refrigerator` |
| `<brand>abccorp</brand>` | `<01>abccorp` |
| `<price>1500</price>` | `<01>1500` |
| `</product>` | |
| `</products>` | |

## 3.4 Handling Tag Attributes

An atomic tag or nested tag can have associated attributes and values. An attribute associated with any tags is used to provide additional information relevant to that tag. A tag can have any number of attributes provided each attribute name is unique

Table 3.4: LXML generated for an XML document with mixed tags

| **XML Format** | **LXML Format** |
|---|---|
| `<invoice>` | |
| `<brand>` | |
| `<name>ABC Corp</name>` | `Invoice` |
| `<model>Series A</model>` | `<00>` |
| `<addr>Delhi</addr>` | `<01>ABC Corp` |
| `</brand>` | `<01>Series A` |
| `<products>` | `<01>Delhi` |
| `<product>` | `<00>` |
| `<name>A01</name>` | `<01>` |
| `<price>1400</price>` | `<02>A01` |
| `<units>15</units>` | `<02>1400` |
| `</product>` | `<02>15` |
| `<product>` | `<01>` |
| `<name>A02</name>` | `<02>A02` |
| `<price>1450</price>` | `<02>1450` |
| `<units>10</units>` | `<02>10` |
| `</product>` | |
| `</products>` | |
| `</invoice>` | |

for a tag. In LXML messages, the attributes can be associated with the tag levels of corresponding tags according to the format **<level attribute1 = value1, attribute2 = value2>** as shown in Table 3.5.

Table 3.5: LXML message for a XML message with attributes

| **XML Format** | **LXML Format** |
|---|---|
| `<products>` | |
| `<product series='A'>` | |
| `<name  grade='A1'>tv</name>` | `products` |
| `<brand>soniscorp</brand>` | `<00 series='A'>` |
| `<price>1000</price>` | `<01 grade='A1'>tv` |
| `</product>` | `<01>soniscorp` |
| `<product  series='B'>` | `<01>1000` |
| `<name grade='A2'>fridge</name>` | `<00 series='B'>` |
| `<brand>abccorp</brand>` | `<01 grade='A2'>fridge` |
| `<price>1500</price>` | `<01>abccorp` |
| `</product>` | `<01>1500` |
| `</products>` | |

## 3.5 LXML Schema

Schemas provide the fundamental building blocks for data organising tasks especially while dealing with bulk amount of data. Document organisation, including sections, paragraphs, lists, and figures, can be described by a document schema. Adherence of a document to its schema is really helpful in document structuring, organizing data, and processing and extracting data at the other end as well.

LXML uses a reference XML as its schema. The LXML messages are created in accordance with this schema. Creation of LXML Schema is a one - time task and provides the details of root tag, atomic tags, repeating container tags, attributes and the overall document structure. Any number of LXML records could be added as per this schema. The LXML Schema will be available at the source end where LXML messages are created and at the destination end where the document is extracted and processed. There will be a LXML Schema defined for each LXML transaction;

it defines the mapping rules associated with it. The tag name given for the root tag should exactly match with that defined in the LXML Schema. The tag name used for atomic and other nested tags is not a concern for transmission but may help in processing and database object creation during data storage.

Table 3.6(a) illustrates a sample invoice XML document that displays the details of products manufactured by a certain organization. This sample XML given is a huge document that holds the details of thousands of products. The LXML message generated corresponding to this sample XML is also displayed as Table 3.6(b). The LXML Schema required for handling LXML messages is as shown in Table 3.6(c). It conveys the overall structure of the document including the details of root tag, container tags, and repetitive tags. Any tag having attributes should also be specified in the LXML Schema. It is very clear that the LXML Schema is not an exact copy of the XML message and is independent of the size or number of records available in the LXML document. LXML Schema is a hierarchical representation of elements contained in the original XML document. LXML Schema surely helps in validating the LXML message at the destination and any mismatch can be easily identified and isolated.

## 3.6 LXML Document Handling

Document handling refers to the collective steps taken to manage the LXML document throughout their life cycle ranging from its creation, transmission, retrieval, storage and ultimately, to its disposal. This involves organizing and managing the document in such a way that it can be easily accessed by the users. The steps in handling an LXML document are as demonstrated in Figure 3.1.

## 3.7 LXML Generation

When handling LXML documents, the first step is to create the LXML document according to the format laid down by the LXML Schema. This step involves defining

Table 3.6: XML message with corresponding LXML message and LXML Schema

| **(a) XML Format** | **(b) LXML** | **(c) LXML Schema** |
|---|---|---|
| `<invoice>` | | |
| `  <brand>` | | |
| `    <name>ABC Corp</name>` | | |
| `    <model>Series A</model>` | `invoice` | |
| `    <addr>Delhi</addr>` | `<00>` | |
| `  </brand>` | `<01>ABC Corp` | `<invoice>` |
| `  <products>` | `<01>Series A` | `   <brand>` |
| `     <product>` | `<01>Delhi` | `     <name></name>` |
| `        <name>A01</name>` | `<00>` | `     <model></model>` |
| `        <price>1400</price>` | `<01>` | `     <addr></ addr>` |
| `        <units>15</units>` | `<02>A01` | `   </brand>` |
| `     </product>` | `<02>1400` | `   <products>` |
| `     <product>` | `<02>15` | `     <product>` |
| `        <name>A02</name>` | `<01>` | `        <name></name>` |
| `        <price>1450</price>` | `<02>A02` | `        <price></price>` |
| `        <units>10</units>` | `<02>1450` | `        <units></units>` |
| `     </product>` | `<02>10` | `     </product>` |
| `     <product>` | `<01>` | `   </products>` |
| `        <name>A0n</name>` | `<02>A0n` | `</invoice>` |
| `        <price>1550</price>` | `<02>1550` | |
| `        <units>40</units>` | `<02>40` | |
| `     </product>` | | |
| `   </products>` | | |
| `</invoice>` | | |

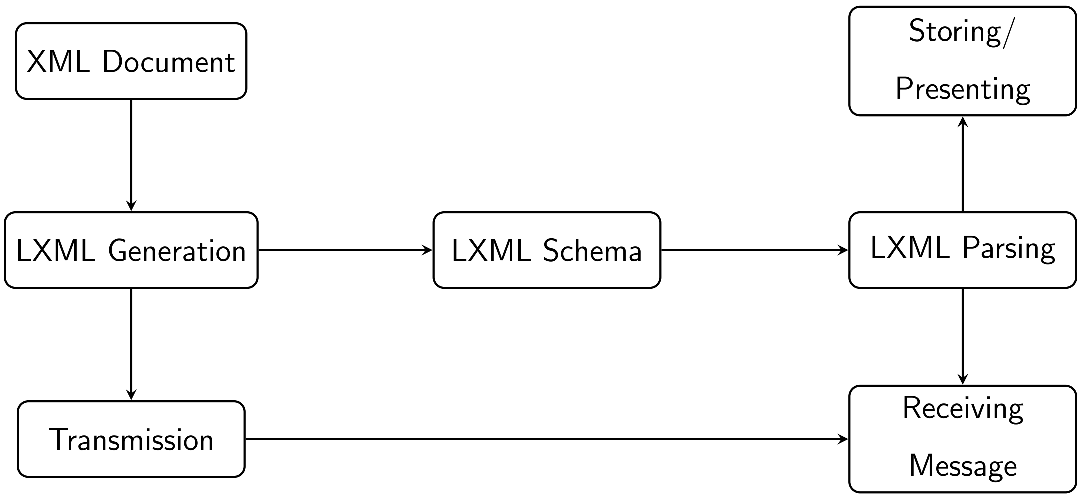

Figure 3.1: Steps in handling LXML document

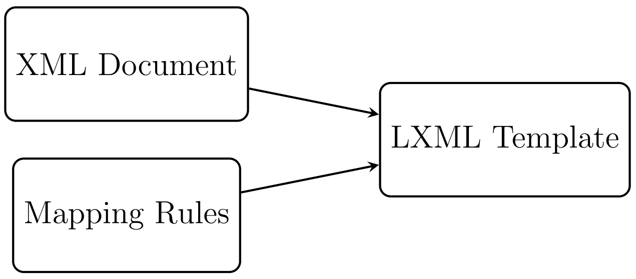

Figure 3.2: Generating LXML document

level based tags and attributes to represent the structure and content of the LXML document. There are two ways to create LXML documents - a LXML document can be generated from an existing XML document. This step is considered as easy as the schema is already in hand. Alternatively, LXML documents can be directly generated in accordance with the LXML Schema.

The LXML document represents data in a hierarchical fashion as in XML. The LXML document is created from an XML document based on a set of mapping rules (Figure 3.2). These mapping rules define the attributes, level numbers in case of nested container tags and repetitive blocks. The mapping rules are defined in the LXML Schema. The LXML thus generated is found to be less verbose than that of the XML documents.

**Algorithm 1:** Algorithm to generate an LXML message

```
Data: Data in plain / XML form
Result: LXML String
Set level number l as 0
if the document has a non-empty root node then
    Add the node to the resultant LXML
end
for each child node T in the XML document do
    if T is a simple node then
        Extract the data from the node and add to the LXML as <level
          No> data
    end
    if T is a container node then
        Increase the level number
        for each child node T in the conatiner node do
            if T is a simple node then
                Extract the data from the node and add to the LXML as
                  <level No> data
            end
        end
        Decrease level number
    end
end
```

The steps to generate an LXML message from an XML is given as Algorithm 1. The root tag name of the XML is retained in LXML message to identify the transaction. The data is extracted directly from simple non-nested tags and added to the LXML message along with its level number. Container tags holds repeating blocks in the XML and such blocks are iterated to retrive the data. The number of iteration required depends on the level of nesting in the XML document. Nested blocks keep data one level lower than its parent node. So each iteration increments

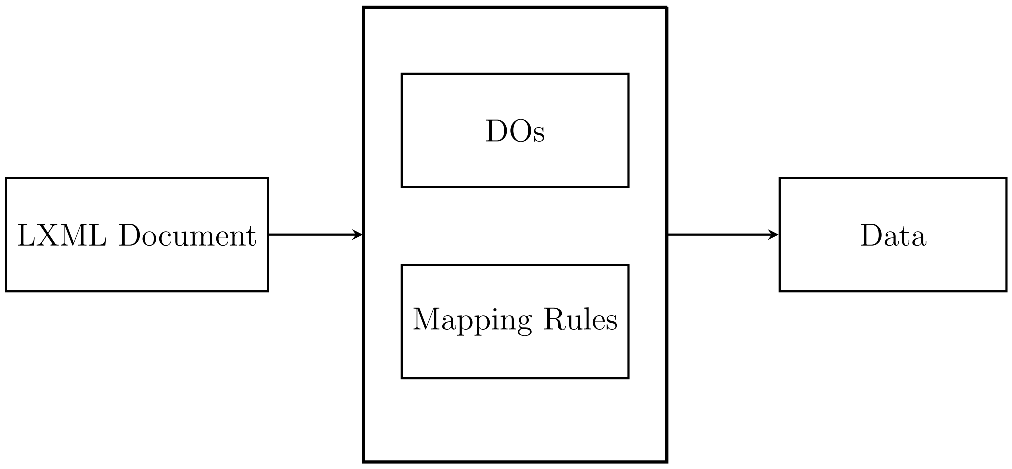

Figure 3.3: Steps in LXML parsing

its level number to indicate the nesting. After processing nested blocks, the level number is decremented.

The complexity of this algorithm depends on the level of nesting used in the XML file. LXML generation of an XML file with only simple tags yields a constant complexity where as the complexity for a file with one level of nesting will be O(n).

## 3.8 LXML Parsing

LXML parsing is the process of analysing the document and extracting information after breaking it into its constituent components. Parsing is essential to migrate the data contained in the document into any desired format or storage.

Though LXML is a format derived from XML, the parsing techniques used in LXML are quite different from XML. In XML there are many parsing techniques such as DOM and SAX which convert the document into a tree - like structure before parsing. This tree structure creation incurs additional overhead in processing and memory requirements. A template based mapping model is proposed for parsing the LXML document. LXML parsing is carried out in two steps: (i) creating data objects (DO), and (ii) mapping (refer Figure 3.3).

Data Object (DO) is a structural pattern that enables us to separate the application / business layer from the persistence layer. Usually, DOs are defined as a class or user defined data type, with data members and member functions corresponding to each tag in the corresponding LXML document. The member functions available in DOs are setter and getter methods for each data member. A sample XML document with its LXML alternative and data binding class is shown in Figure 3.7.

Table 3.7: An XML message, its LXML equivalent and Data Object

| **XML Format** | **LXML Format** | **Data Binding** |
|---|---|---|
| `<product>`<br>   `<name>television</name>`<br>   `<manufa>sonis</manufa>`<br>   `<price>100</price>`<br>`</product>` | `product`<br>`<00>television`<br>`<00>sonis`<br>`<00>1000` | `Class Product {`<br>  `String name`<br>  `String manufa`<br>  `int price`<br>`}` |

A single LXML document may map to multiple DOs depending on the business needs and document structure. A sample LXML document and its data binding class are shown in Figure 3.8.

The terms mapping or binding refer to the process of assigning LXML tag values to corresponding data models utilizing the set of binding rules defined earlier. Since the binding rules are defined much earlier, a static data binding approach is used. The tag values are assigned to the data members of the DOs corresponding to the specification given in the LXML Schema. The mapping rules given in the LXML Schema is the basis for data extraction as shown in Figure 3.3. The splitting of the document is done based on level numbers. LXML data with same level numbers can be split and dynamically assigned to the DO easily. An LXML document with repeating groups will generate an array of DOs. The DOs later can be readily stored to a relational database or presented to the user interface as required.

Table 3.8: Data binding of a sample LXML document

| **XML Format** | **LXML Format** | **Data Binding** |
|---|---|---|
| `<invoice>`<br>`<brand>`<br>`<name>ABC Corp</name>`<br>`<model>Series A</model>`<br>`<addr>Delhi</addr>`<br>`</brand>`<br>`<products>`<br>`<product>`<br>`<name>A01</name>`<br>`<price>1400</price>`<br>`<units>15</units>`<br>`</product>`<br>`<product>`<br>`<name>A02</name>`<br>`<price>1450</price>`<br>`<units>10</units>`<br>`</product>`<br>`</products>`<br>`</invoice>` | `Invoice`<br>`<00>`<br>`<01>ABC Corp`<br>`<01>Series A`<br>`<01>Delhi`<br>`<00>`<br>`<01>`<br>`<02>A01`<br>`<02>1400`<br>`<02>15`<br>`<01>`<br>`<02>A02`<br>`<02>1450`<br>`<02>10` | `Class Brand{`<br>`String mname;`<br>`String model;`<br>`String address;`<br>`}`<br><br>`Class Product{`<br>`String pname;`<br>`float price;`<br>`int units;`<br>`}` |

## 3.9 Performance Evaluation

In resource constrained networks such as mobile messaging environments, an enhanced messaging format is primarily required to boost data transmission efficiency [152]. Such an enhanced, lightweight data interchange format provides many benefits:

(a) Reduced data size: An efficient data interchange format reduces the amount of data transmitted between the communicating devices. A reduction in data size reduces the demand for resources such as bandwidth, memory and energy which are very critical in resource constrained environments.

(b) Fast processing: A lightweight, less resource intensive format can be processed more quickly than its heavy weight alternative.

(c) Improved reliability: The likelihood of error and corruption during transmission will be less in an efficient data interchange format. This enhances the data integrity and reliability.

(d) Higher compatibility: An efficient data interchange format can be designed to be compatible with a wider range of devices and systems, which is highly appreciated in environments where interoperability is crucial.

This section discusses the results of the experiments carried out to demonstrate the performance of the proposed format. The performance of the LXML format is evaluated using the following performance parameters: verbosity, content density, parsing time, serialization & deserialization time, marshalling & unmarshalling time, and transmission time [153][154]. The values obtained are compared with that of XML and JSON, the most prominent data interchange formats, to emphasize the potential of the proposed format.

To consider different aspects in experimentation XML, JSON and LXML documents of varying sizes and object counts are considered. The documents are classified into four categories depending on the size as shown in Table 3.9, based on the discussion in [75].

Table 3.9: Dataset for experiments

| **Dataset** | **File Size** | **No. of Records** |
|---|---|---|
| Very small | in KBs | 1 - 20 |
| Small | Less than 0.5 MB | 100 - 1000 |
| Medium | 0.5 MB to 1 MB | 1000 - 20000 |
| Large | Above 1 MB | 100000 - 150000 |

Table 3.10: Verbosity of XML, JSON and LXML documents for varying file sizes

| **Format** | **Very small (Size in KB)** | **Small(Size in KB)** | **Medium (Size in KB)** | **Large (Size in KB)** |
|---|---|---|---|---|
| XML | 4.57 | 376 | 603 | 2190 |
| JSON | 3.90 | 314 | 689 | 2307 |
| LXML | 2.52 | 209 | 309 | 1342 |

### 3.9.1 Experimental Setup

The environment to test the proposed format is configured using a laptop (with core i3 processor, 4GB RAM, and Microsoft Windows 10 as operating system) and a mobile phone (Android version 9). The laptop has JDK version 1.8 installed. JABX toolkit and Jackson API [155] are used for marshalling and unmarshalling XML and JSON objects. SoapUI tool is used for simulating servers. The experiment is carried out with sample files of varying sizes (Table 3.10). Each experiment is repeated at least three times and average values are considered to ensure the accuracy and the reliability of the simulation.

#### Verbosity

Verbosity refers to the level of detail or amount of information present in the document. It is always a factor of document size. When the verbosity increases, the transmission overhead increases proportionally. So, a less verbose format is suitable for data exchange especially in resource constrained environments.

The verbosity of XML, JSON, and LXML documents for various categories of dataset are displayed in Table 3.10. The result is plotted as a graph by taking file sizes on the Y axis and the X axis representing XML, JSON, and LXML formats as shown in Figures 3.4, 3.5 and 3.6, for small, medium and large file sizes respectively.

It is clear that LXML is the least verbose representation and XML is the most verbose among the three formats. JSON reported an advantage over XML in small files; as and when the file size increases, the performance of JSON is more or less

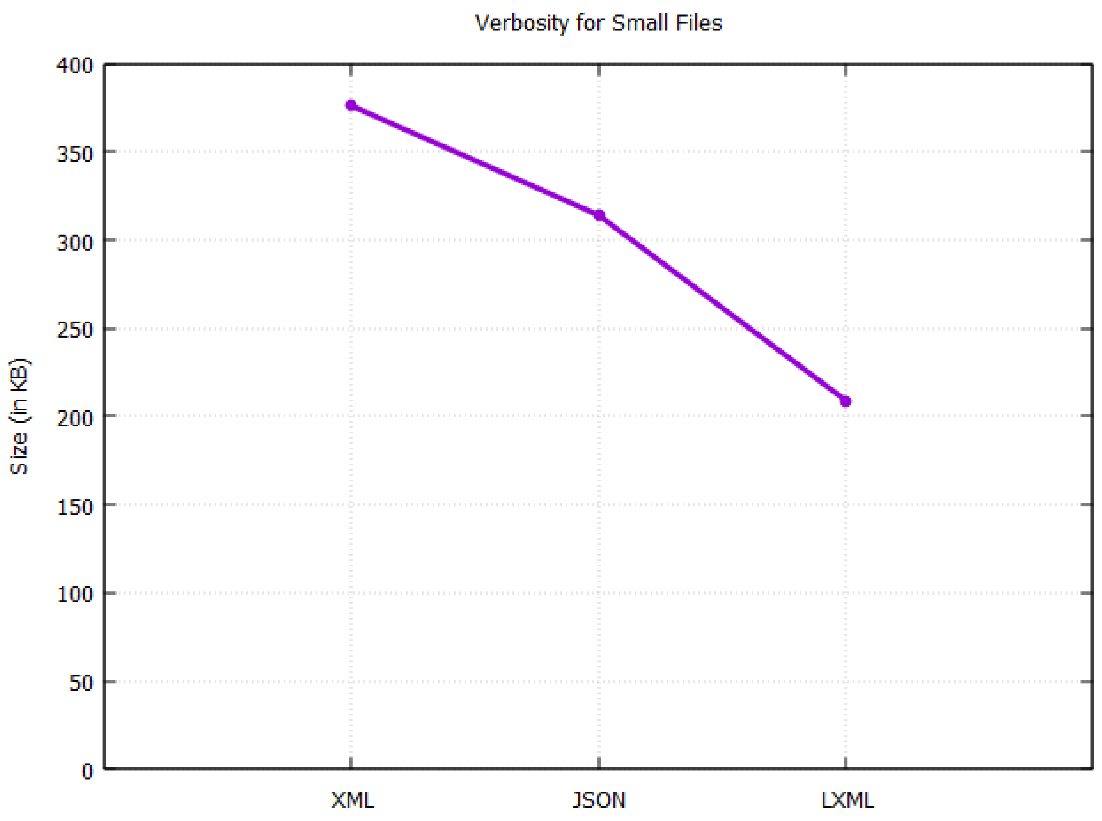

Figure 3.4: Verbosity comparison (Y axis) of XML, JSON, and LXML (X axis) for small category files

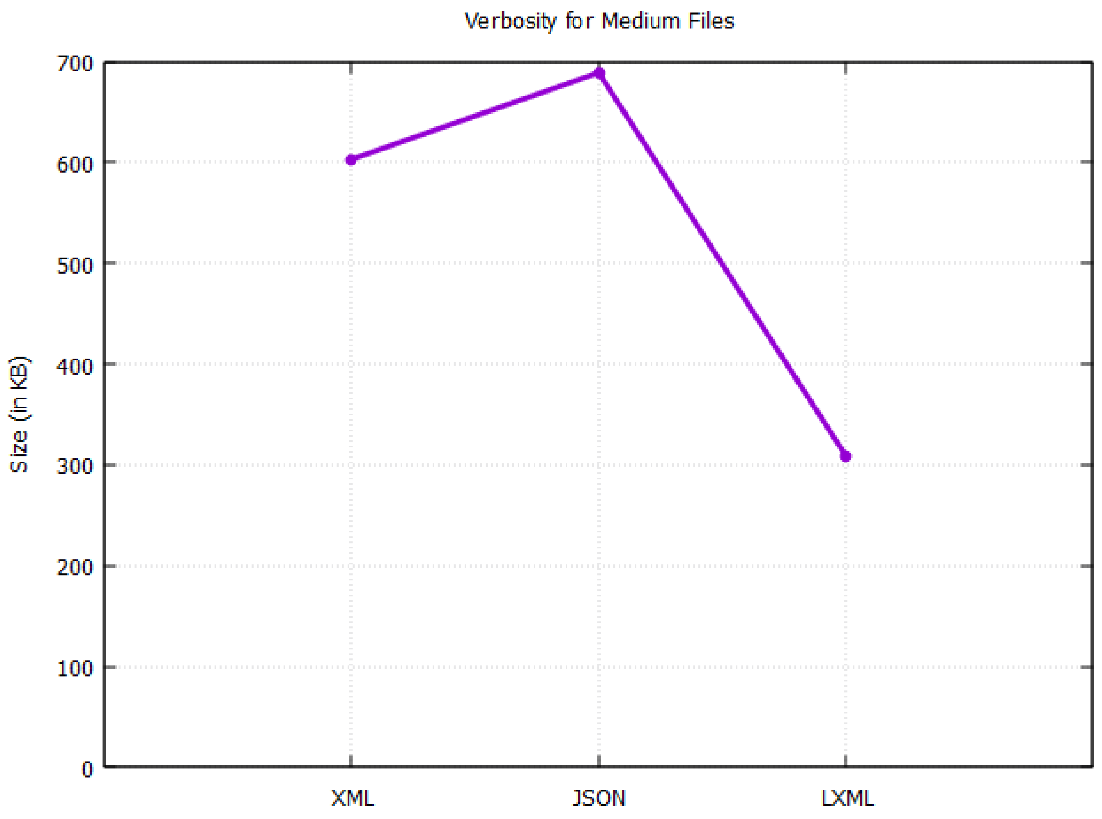

Figure 3.5: Verbosity comparison (Y axis) of XML, JSON, and LXML (X axis) for medium category files

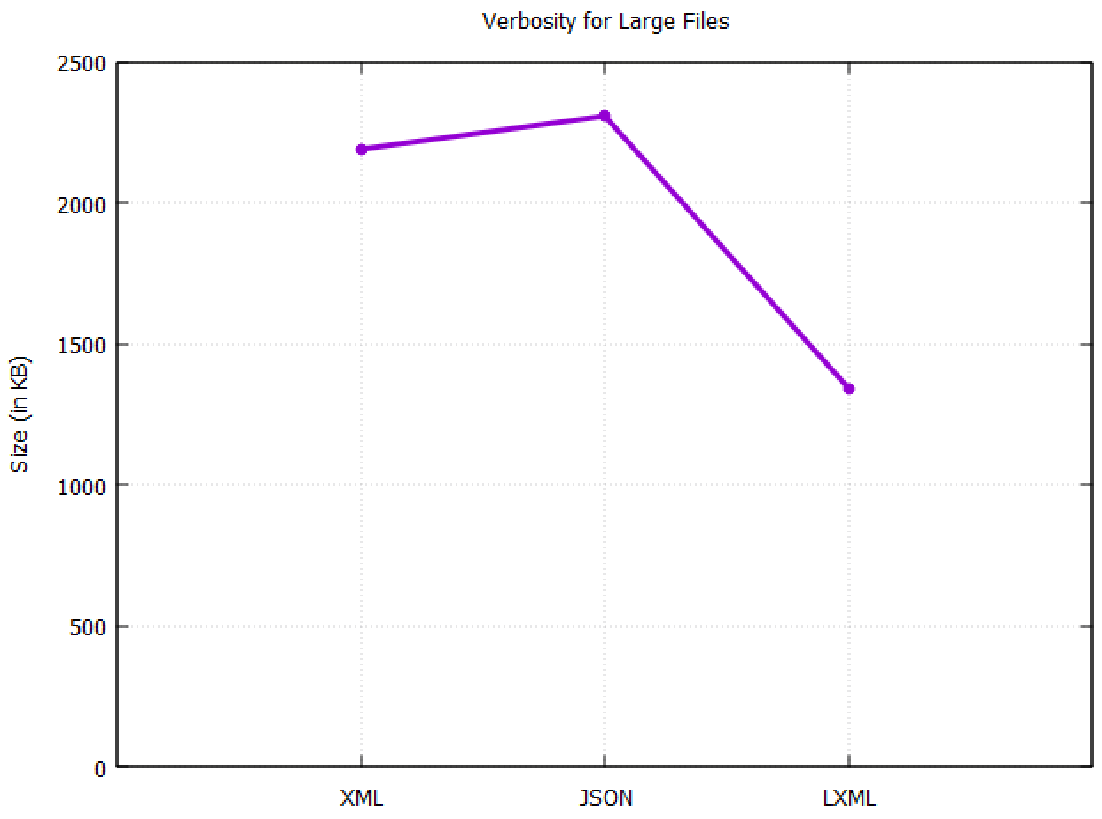

Figure 3.6: Verbosity comparison (Y axis) of XML, JSON, and LXML (X axis) for large category files

equal to that of XML.

While discussing the document verbosity, another factor to be considered is the size of the schema. As the schema defines the structure and organization of the document, its size also matters. JSON does not support any schema. All the datasets mentioned in Table 3.9 have the same LXML Schema. Verbosity of XML Schema and LXML Schema is tabulated in Table 3.11. As the schema is independent of the number of objects present in the document, its size remains the same for all samples. It can be inferred that the LXML Schema is less verbose than XML Schema.

Table 3.11: Verbosity comparison of XML and LXML Schema

| **Format** | **Verbosity** |
|---|---|
| XML Schema | 1.34 KB |
| LXML Schema | 246 Bytes |

### Content Density

Content density is the ratio of total amount of data available in the document to the total size of the document [153][156]. Content density of any data interchange format designed to transmit data in a resource constrained environment is critical due to many reasons. A format with higher content density allows transmitting more data in a short period of time reducing resource required for data exchange. Content density is also considered as important in storage and retrieval of data especially while dealing with bulk amounts of data. In short, a data interchange format with a high content density can improve the speed, efficiency, and performance of data exchange and processing; it also reduces the storage requirements and costs.

A format with content density 1 is considered as more efficient. Adding more metadata pushes the content density towards 0, making it a verbose format. Table 3.12 displays the content density values of XML, JSON and LXML formats for all categories of dataset. A graphical representation of the content densities for various data interchange formats are displayed in Figure 3.7.

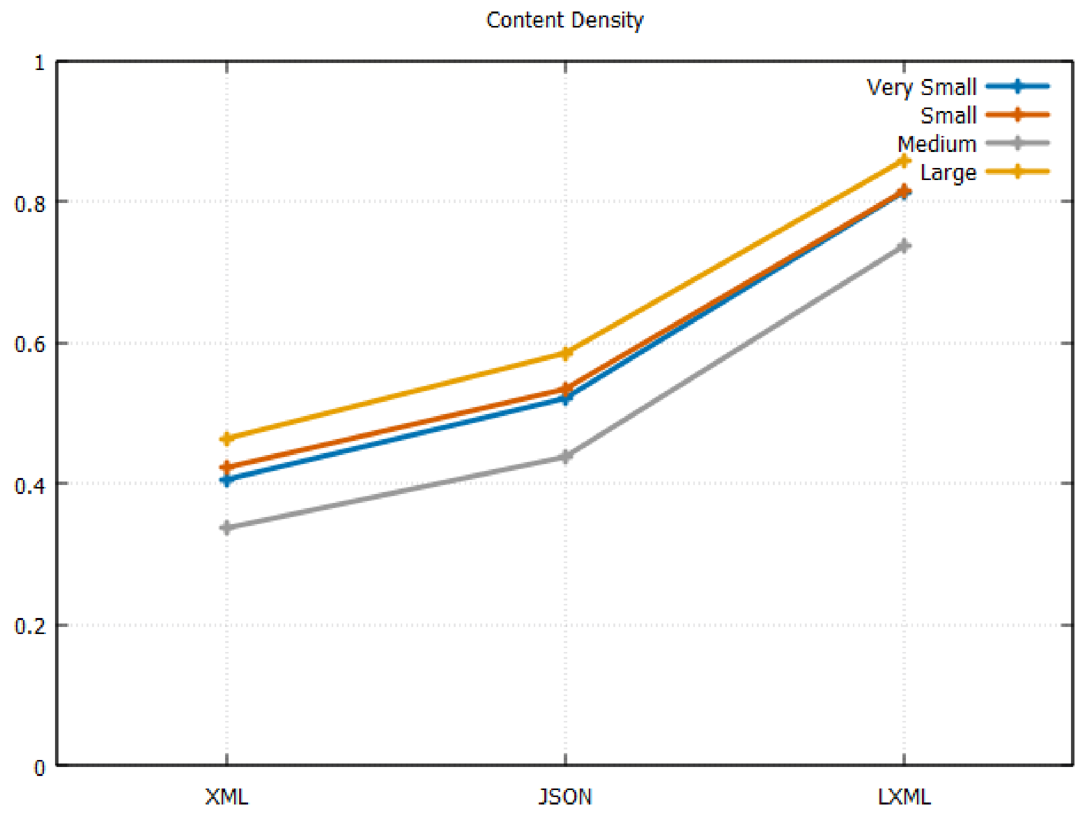

Figure 3.7: Comparison of content densities (Y axis) of XML, JSON, and LXML (X axis) documents with varying sizes

Table 3.12: Comparison of content density for XML, JSON, and LXML documents

| Dataset | XML | JSON | LXML |
|---|---|---|---|
| Very small | 0.406 | 0.521 | 0.814 |
| Small | 0.423 | 0.534 | 0.815 |
| Medium | 0.337 | 0.438 | 0.767 |
| Large | 0.464 | 0.585 | 0.859 |

It is clear that the content density values of LXML is close to 1 for all the data sets. XML is the most non-compact format and JSON has a slight advantage over XML. It can be inferred that data content in an XML document is much less than 50% of the total document size, whereas in the case of LXML, it is above 80%.

**Parsing Time**

Parsing refers to the process of extracting relevant data from a document. In addition to data extraction, parsing is required for validating and transforming the document to a different format as well. Processing speed is more important in a wireless environment as mobile devices have less processing power compared to static devices [55][157].

XML uses different technologies to parse the document. Commonly used parsing approaches in XML include DOM, SAX, XPath and Pull parsers [74][158]. The DOM parser converts the XML document to an in - memory object tree representation and keeps the entire tree in the memory for processing. SAX parsers are event based parsing techniques that processes the XML document sequentially; the document is converted into a series of identified events. Though SAX parsing is fast and memory efficient, it requires more coding effort than DOM parsing. XPath is a query language that allows selection of specific elements in an XML document.

Performance of parsing the XML and JSON depends on the technology used. Parsing an XML document using JavaScript is more efficient than JSON; but in the case of querying, JSON has an edge over XML [154]. In this experiment, DOM parsers are used for XML parsing due to its simplicity and intuitive nature. DOM - like and SAX - like parsing approaches are available for parsing JSON documents. For this experiment, Java API JsonParser [79] is used to parse JSON. As LXML uses object based mapping techniques for parsing documents, a standalone Java program is written to parse LXML documents. The average time consumed (in ms) in parsing XML, JSON and LXML documents consisting of different record count is as shown in Table 3.13.

Table 3.13: Parsing time for XML, JSON, and LXML documents of varying number of records

| | **Average time consumed (in ms)** | | |
|---|---|---|---|
| **No of Records** | **XML** | **JSON** | **LXML** |
| 1000 | 129 | 90 | 79 |
| 5000 | 534 | 326 | 261 |
| 10000 | 1320 | 779 | 731 |

The results are graphically presented in Figure 3.8 (Parsing time in the Y axis). It can be inferred that LXML parsing technique consumes less time compared to that of XML and JSON. This is due to the fact that LXML does not have processing overhead in terms of processor time and memory.

### Serialization and Deserialization Time

Serialization is the process of converting an object to a format that can be easily transmitted over the network. Usually this process converts the object into a stream of bytes. Deserialization is the process of converting the serialized data back into an object. This is performed at the receiving end where the stream of bytes is converted back to objects.

Serialization and deserialization are important concepts in networking as it specifies how easily the data is transmitted or stored in standard format [159][160]. Serialization and deserialization operations performed on data interchange formats such as XML and JSON objects. The execution speed in serializing and deserializing

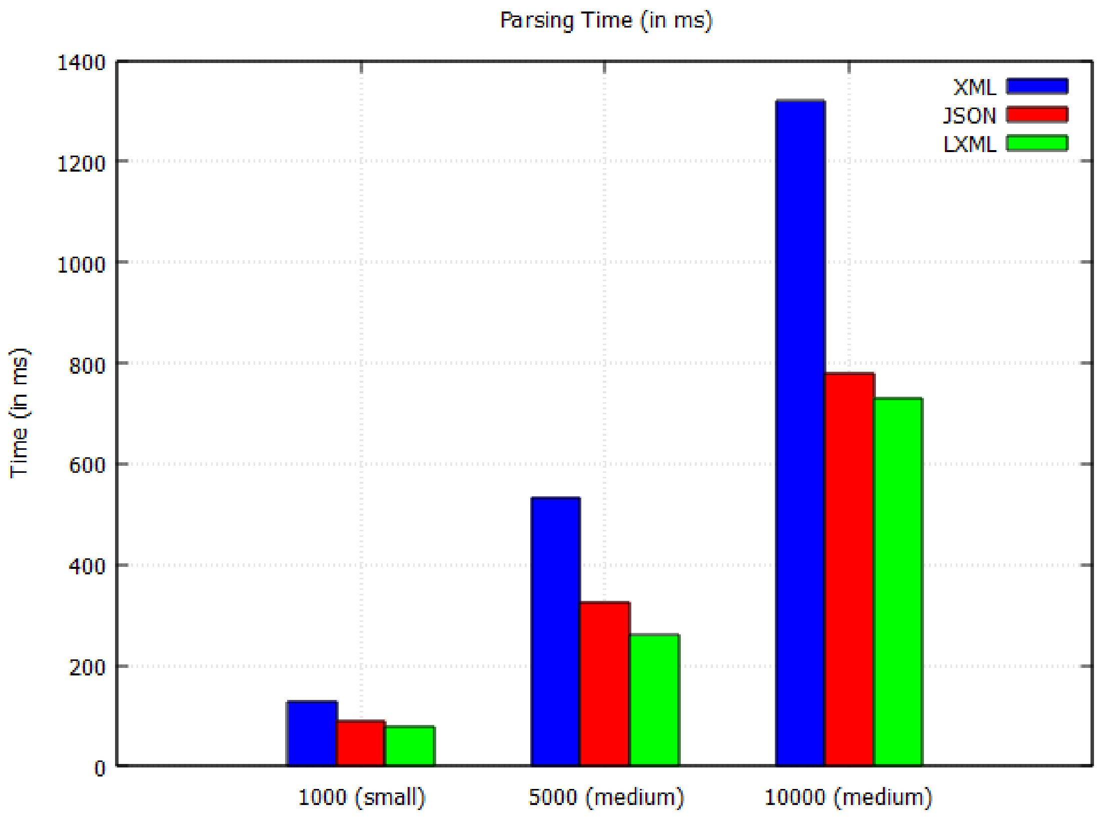

Figure 3.8: Comparison of parsing time (Y axis) for XML, JSON, and LXML (X axis) documents of varying number of records

objects is an important performance criterion in data interchange formats [40]. This performance evaluation is carried out using XStream [161] and JSON libraries [162], for XML and JSON, respectively. The time required to serialize the objects at the serving side is positively correlated to the number of objects [81].

Table 3.14: Serialization and deserialization time (in ms) for XML, JSON and LXML

| | | **Average time elapsed (in ms)** | | |
|---|---|---|---|---|
| **Process** | **Format** | **Small** | **Medium** | **Large** |
| Serialization (average time consumed in ms) | XML | 2.860 | 3465 | 7214 |
| | JSON | 3.146 | 5356 | 10812 |
| | LXML | 2.421 | 2954 | 5824 |
| Deserialization (average time consumed in ms) | XML | 5.320 | 4654 | 9405 |
| | JSON | 4.405 | 9476 | 18423 |
| | LXML | 4.224 | 4320 | 8640 |

The average time consumed (in ms) for serializing and deserializing XML, JSON and LXML documents for varying record sizes are tabulated in Table 3.14. It can be observed that the proposed LXML format has an advantage over XML and JSON in terms of time required for serializing and deserializing for documents in the small category (Figure 3.9). In medium and large files, LXML has reported a narrow edge over XML in deserialization (Figure 3.10 and 3.11). But it has a clear performance advantage over JSON. As the serialization and deserialization process directly impact the transmission of the document, clearly LXML has an edge over other formats.

### Marshalling and Unmarshalling Time

Marshalling and unmarshalling are important processes applied in data exchange between heterogeneous systems or programs. Marshalling is the process of converting an object to a format acceptable for communication such as XML or JSON.

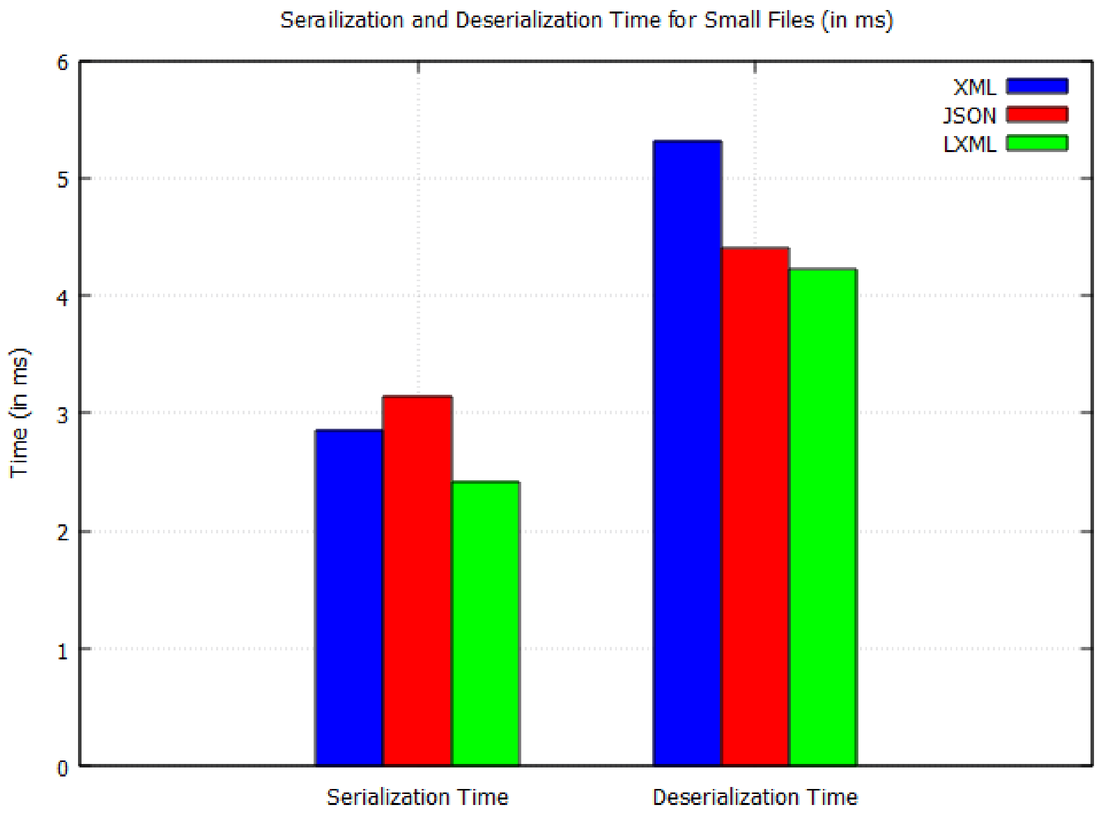

Figure 3.9: Serialization and deserialization time in ms (Y axis) for XML, JSON, and LXML (X axis) documents of small category

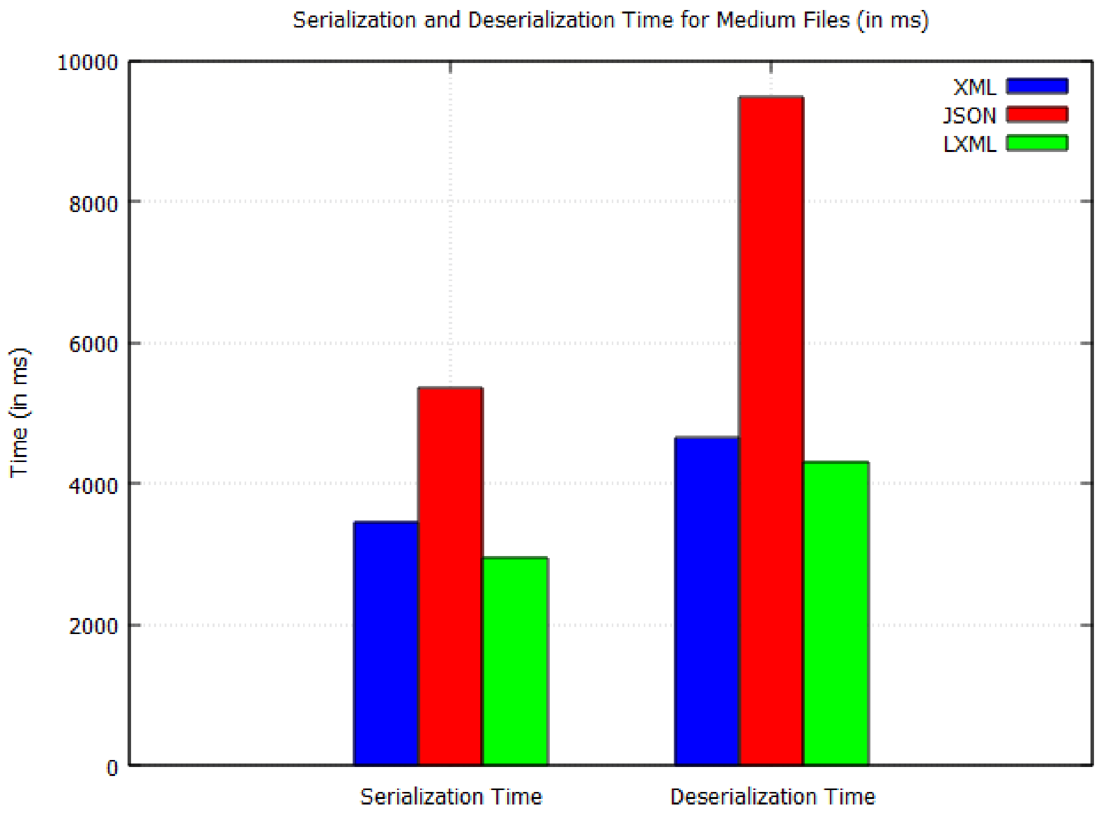

Figure 3.10: Serialization and deserialization time in ms (Y axis) for XML, JSON, and LXML (X axis) documents of medium category

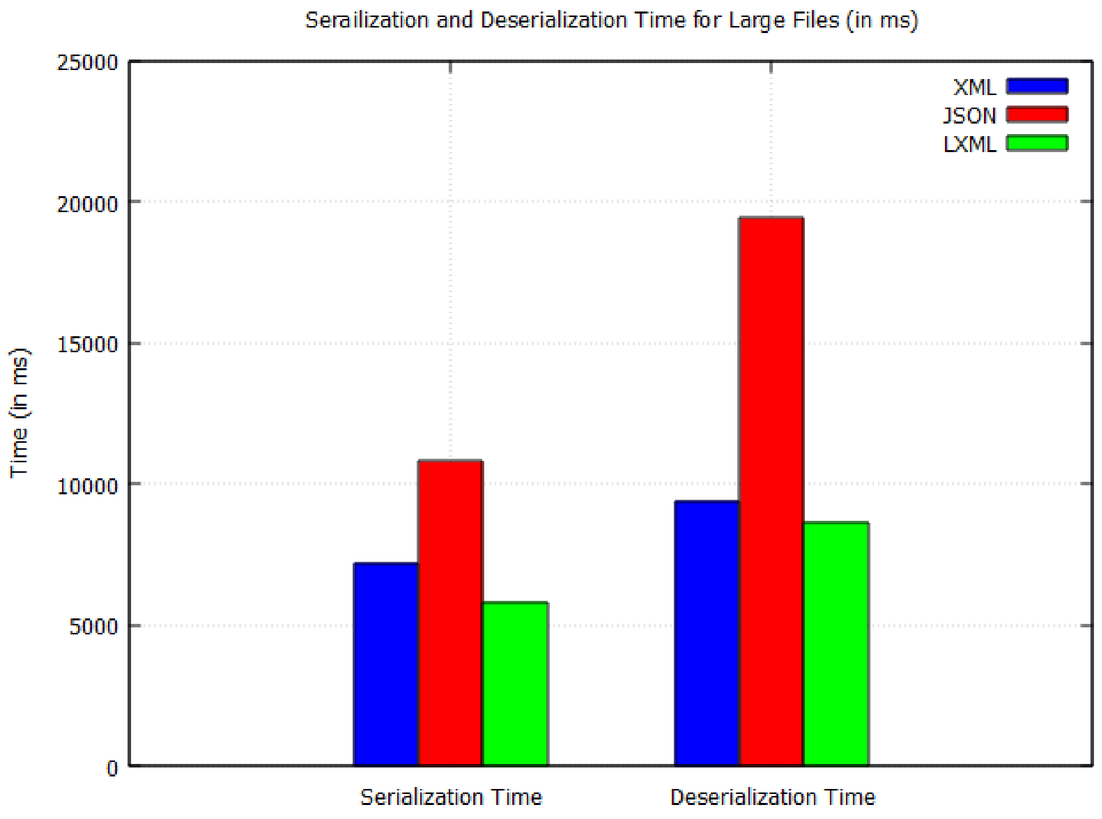

Figure 3.11: Serialization and deserialization time in ms (Y axis) for XML, JSON, and LXML (X axis) documents of large category

This process involves taking an object and generating a document out of it. Unmarshalling is the reverse operation where the object is recreated. This involves extracting the data from the document and creating the structured objects.

The time and memory consumed for marshalling and unmarshalling is considered as a performance criterion in data transfer [163]. High marshalling throughput with less latency and compact memory usage is highly appreciated for achieving higher overall performance.

The XML and JSON objects are serialized before performing marshalling. The JABX toolkit and Jackson API are used for marshalling and unmarshalling these objects [155][164]. Marshalling and unmarshalling LXML DOs are necessary while processing and transmitting documents. In LXML the marshalling latency is the sum of the time consumed for LXML serialization and DO mapping. Memory footprints and execution time required for marshalling and unmarshalling various data formats for small dataset is tabulated in Table 3.15 and the results are plotted in Figure 3.12 and 3.13.

Table 3.15: Marshalling and unmarshalling time of XML, JSON and LXML

| | **XML** | **JSON** | **LXML** |
|---|---|---|---|
| File Size in KB (small) | 8 | 3.9 | 2.52 |
| Memory in KB (marshal) | 533 | 91.57 | 73.24 |
| Memory in KB (Unmarshal) | 143 | 121.38 | 96.34 |
| Execution Time in millisec (marshal) | 0.024 | 0.0108 | 0.0102 |
| Execution Time in millisec (Unmarshal) | 0.044 | 0.0141 | 0.0143 |

It can be observed that the memory footprints and execution time for marshalling and unmarshalling of LXML has a narrow edge over JSON and the performance is far better than XML.

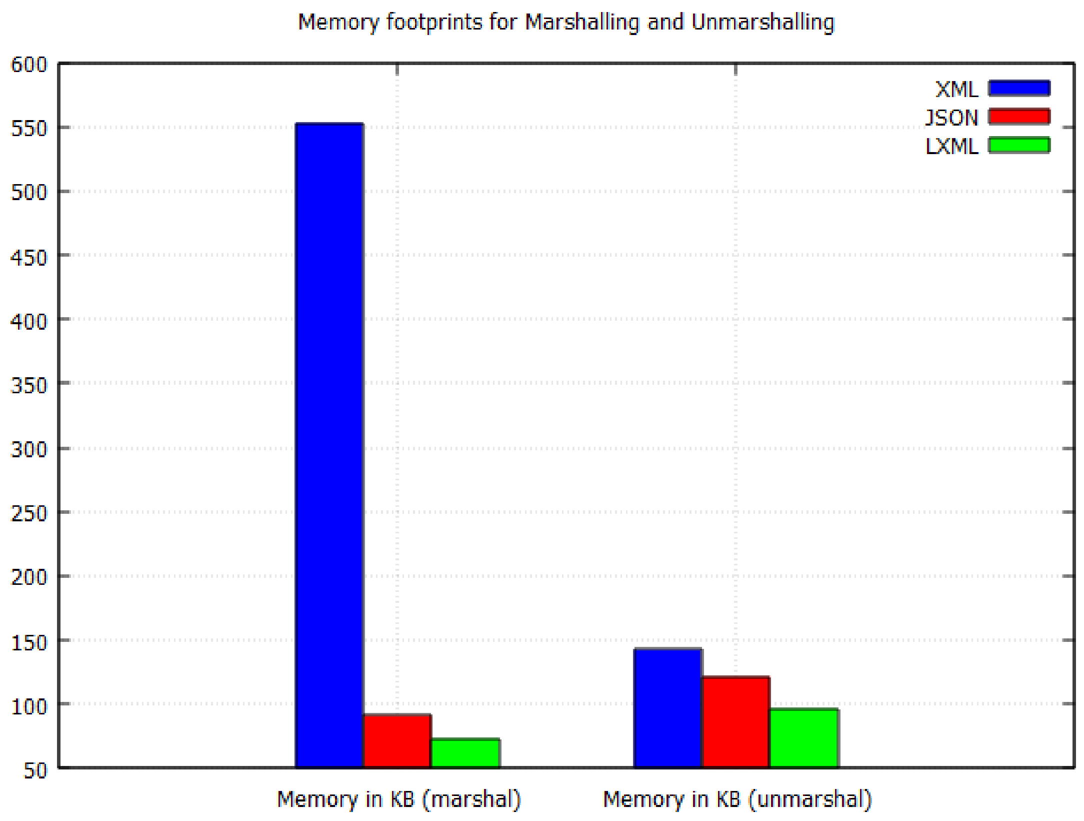

Figure 3.12: Memory footprints (Y axis) for marshalling and unmarshalling for XML, JSON, and LXML (X axis) documents of small category

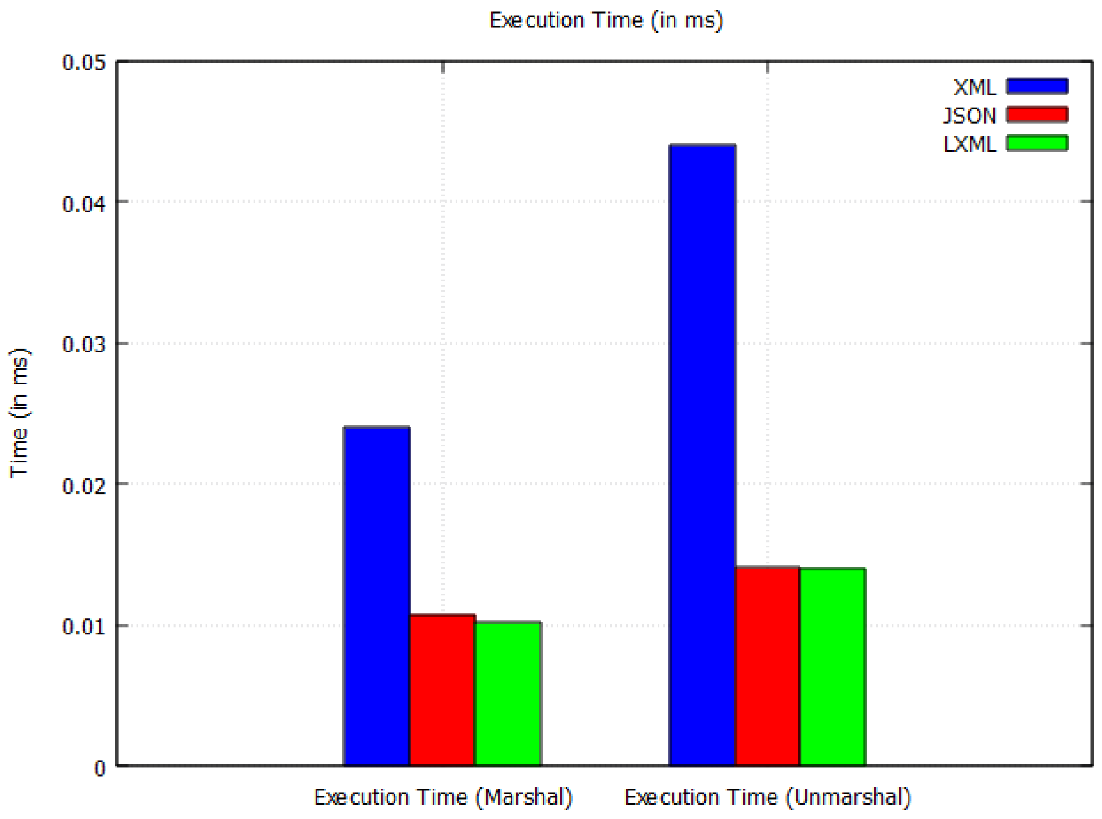

Figure 3.13: Execution time (Y axis) for marshalling and unmarshalling for XML, JSON, and LXML (X axis) documents of small category

**Transmission Time**

The transmission time is the time taken for a node to send a document to another node. The transmission time depends on several factors such as the size of the document, the transmission power, the distance between the nodes, and the modulation technique used. Minimizing the transmission time can reduce energy consumption and improve the network efficiency. Transmission time in Wireless networks can be calculated using the following formula [165].

$$\text{Transmission Time} = \text{Size}/\text{K} + \text{D}/\text{C} \tag{3.1}$$

K is calculated using the equation 3.2.

$$\text{K} = (2 * \text{Size}) \ / \ \text{R} \tag{3.2}$$

Size is the packet size in bits, R is bit rate (in bps), and D is the distance between nodes in metres, and C is the velocity of light for wireless communication (in m/s).

The transmission time for various messaging formats is calculated by simulating a network with two nodes 100 KMs apart. Data of different sizes is being sent assuming the speed of signal as 1, 00, 000 KM per sec. The transmission time is tabulated assuming there is no propagation delay and the rate of data transfer is 1 Mbps.

Time for transmission of XML, JSON and LXML documents using datasets of small and medium categories, respectively, are tabulated in Table 3.16 and is graphically presented in Figure 3.14. It is evident that LXML consumes the least time compared to XML and JSON.

## 3.10 Applications

The proposed LXML data interchange format can be utilized in diverse fields and applications. Few of them are listed below.

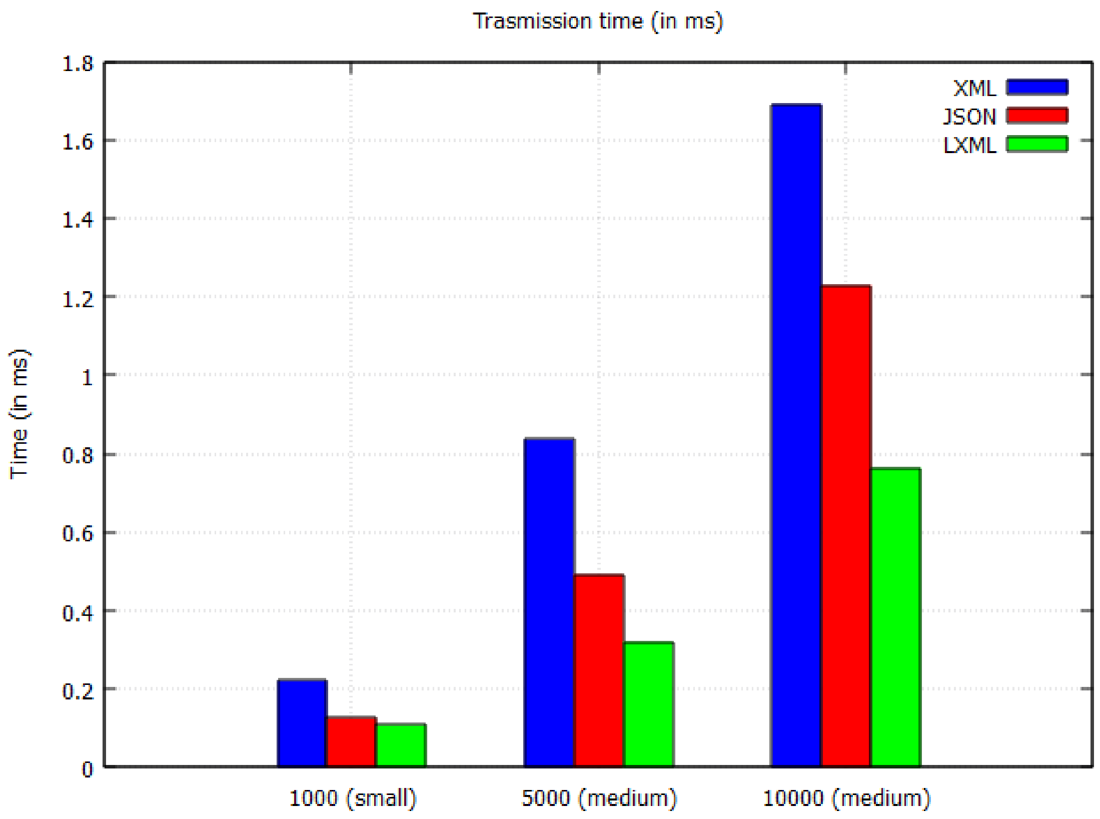

Figure 3.14: Transmission time (Y axis) for XML, JSON and LXML (X axis) documents of small and medium categories

Table 3.16: Transmission time (in ms) for XML, JSON and LXML documents of small and medium categories

| Format | No. of Records | Transmission Time (ms) |
|---|---|---|
| XML | 1000 (Small) | 0.233 |
| | 5000 (Medium) | 0.837 |
| | 10000 (Medium) | 1.690 |
| JSON | 1000 (Small) | 0.128 |
| | 5000 (Medium) | 0.492 |
| | 10000 (Medium) | 1.231 |
| LXML | 1000 (Small) | 0.0109 |
| | 5000 (Medium) | 0.32 |
| | 10000 (Medium) | 0.764 |

(i) LXML is a versatile solution for facilitating seamless data interchange across diverse systems, applications, and platforms, effectively structuring and arranging data for seamless sharing and accessibility.

(ii) LXML can be used for efficiently transferring and presenting extensive product information in dynamic websites and web APIs, enabling seamless data transmission and dynamic filtering based on user interactions.

(iii) LXML is the correct choice for facilitating data exchange and communication within mobile applications due to the inherent constraints of wireless mobile platforms and the need to handle substantial volumes of data across various industries such as healthcare, ecommerce, and finance.

(iv) LXML format can be preferred over conventional verbose formats for managing and transmitting massive datasets in big data and analytics due to its superior performance and efficiency.

(v) LXML format holds a distinct advantage over traditional data interchange formats in IoT environments due to its efficiency and ability to facilitate communication and operation within resource constraints.

## 3.11 Summary

XML and JSON are the two prominent data interchange formats that are human readable. Despite their advantages, they have several performance overheads in transmission and processing, especially in resource constrained wireless mobile networks. These overheads accelerate the energy consumption and drain the mobile battery as well. An alternative data interchange format called LXML is proposed which is lightweight, human readable, platform neutral, schema aware, language neutral and extensible. The performance of the model is compared with XML and JSON using six performance parameters. It is found that the proposed format has an advantage in verbosity (40 - 48% and 30 - 40%, when compared with XML and JSON, respectively). The LXML Schema is 4.6 times less verbose than XML Schema. The content density of LXML averages to 80 - 85% of the document size, which is far better than XML and JSON. LXML also has a considerable advantage in parsing time, serialization & deserialization, marshalling & unmarshalling time, and transmission time over XML and JSON (refer Table 3.17). Thus the new model has the potential to be considered as an alternative data interchange format over others, especially in resource constrained networks.

Table 3.17: LXML: Objectives Vs Accomplishments

| **Criterion** | **XML** | **JSON** | **LXML** |
|---|---|---|---|
| Verbose | High | Medium | Low (40 - 48% and 30 - 40% advantage for LXML over XML and JSON, respectively) |
| Schema aware | Yes | No | Yes, with less verbosity (80 - 90% advantage for LXML over XML) |

| | | | |
|---|---|---|---|
| Content density | Low | Medium | Very high (85 - 120%, 48 - 75% advantage for LXML over XML and JSON, respectively) |
| Parsing time | High | Medium | Low (38 - 51% and 6 - 20% advantage of LXML over XML and JSON, respectively) |
| Serialization & deserialization time | High | High | Low (Serialization: LXML has an advantage of 14 - 24% over XML and 23 - 46% over JSON. Deserialization: LXML has an advantage of 7 - 20% over XML and 10 - 55% over JSON) |
| Marshalling & unmarshalling time | High | Low | Low (56 - 68% and 2 - 6% advantage for LXML over XML and JSON) |
| Transmission time | High | Medium | Low (51 - 61% and 15 - 40% advantage for LXML over XML and JSON, respectively) |

Chapter 4

# Layered Architecture for the Proposed Transaction Model

## 4.1 Introduction

Wireless mobile environment is distributed in nature with multiple interconnected components processing tasks to achieve higher data availability, parallelism and scalability. The components may share and exchange data and coordinate various tasks. So, communication between the components and accessing data independent of their location is crucial. In such an environment, an application running on a handheld device needs to access data available in a CRM, server or any other shared system.

For the reliable communication of data between applications, a transaction model is inevitable. A transaction model in a distributed environment describes the way communication is organized and managed between the participating components within the system. Such a model helps to exchange information without focusing on the inherent complexities of data communication.

Designing a messaging system for a distributed environment involving mobile devices is a challenging task due to the heterogeneous and inconsistent nature of the environment. Mobile applications make use of web services for exchange of information. In such cases, the devices act as mobile client systems. Though XML and JSON are the most widely used and universally accepted formats for structured data transmissions, these formats have been forcibly received by the mobile community in spite of their verbose nature and considerably heavy processing requirements. These requirements adversely affect the performance in the constrained mobile wireless environment, due to the factors such as limited computing, storage and energy capacities of mobile devices, and the slow and expensive nature of the wireless network connections.

The proposed transaction model architecture is designed to overcome the limitations of the existing messaging models in wireless mobile environments. Analysing the existing models, it is clear that there is ample scope for performance improvements in the areas such as processing interfaces, serialization, message size and transfer protocols. Existing transaction models can also be enhanced with more precise data representation formats like LXML. Since the processing interfaces, serialization and protocols depend on the device configurations and capabilities, the most promising solution is to reduce the bulkiness of data transportation. This step will ensure a reverberating effect on all phases concerned with message exchange.

## 4.2 Proposed Architecture

The proposed model is a distributed multi - tier system based on the prominent XML / HTML based agent communication model. This agent based model support distributed, client - server based message exchange using XML. The proposed model opts for the concise LXML format over verbose XML, streamlining data exchange. The model is a three layered architecture as shown in the Figure 4.1. The layers of the proposed model are: (i) client agent (Layer 1), (ii) middleware component (Layer 2), and (iii) host server (Layer 3).

### 4.2.1 Client Agent

The client agent resides in the portable mobile unit and deals with client side activities. The client agent is responsible for communicating with the gateway interface.

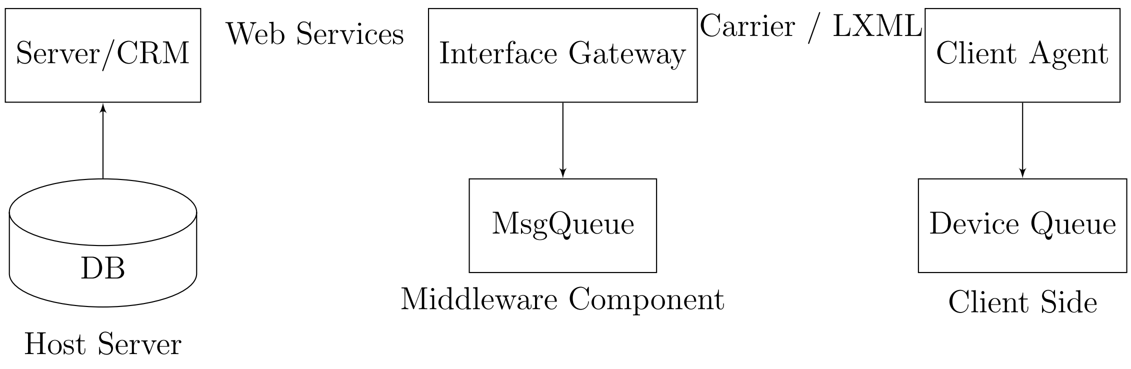

Figure 4.1: Block diagram of the proposed transaction model

It updates the connectivity status to the device through registration packets. Once connectivity is established, it is responsible for generating and transmitting LXML messages out of user interactions in the application user interface. Each message is stored in its internal queuing system until it gets confirmation from the gateway. On the other hand, the client system accepts the messages received from the gateway interface and acknowledges its receipt. Later the LXML messages are parsed and extracted.

### 4.2.2 Middleware Component

The data transfer between the handheld devices and the server is directed through a middleware gateway interface system based on the Service Oriented Architecture. The component acts as an intermediary and is responsible for resolving the differences between the devices and host layer. Usually, the middleware is hosted in a powerful dedicated machine or as in-house in the host server.

The middleware component is composed of two subcomponents: the gateway and the message queue. The gateway interface is the component responsible for device registration and status tracking; it establishes communication with the handheld devices. It accepts message packets containing LXML messages from the devices and converts it to a format acceptable to the host servers. Similarly, the messages from the host server are received by the gateway interface and converted to LXML messages before transmitting to the devices. To ensure the reliability, the gateway interface acknowledges the receipts of messages from both host and devices. Since the messages can be sent and received as packets, the proposed model supports pagination - a message or data item of considerably large size can be conveniently segmented into packets of lesser size and transmitted.

In case a device is not available for data transmission, the message is stored in the message queue available in the middleware system. Such messages are transmitted and the queue is cleared once network connectivity is regained. Hence the measures for disconnectedness is also incorporated in the proposed model.

### 4.2.3 The Host Server

The host server constitutes the backend data layer that is responsible for storing and managing data in a centralized manner. It handles data storage, satisfies retrieval and manipulation of data based on the requests received from the client system through the middleware layer. It handles such requests ensuring data integrity, consistency and security.

Data can be stored in databases or any other data storage systems. Business applications make use of vital data repositories such as Enterprise Resource Planning (ERP) and Customer Relationship Management (CRM) to run and manage their business. While CRM software is responsible for managing the customer interactions with the business, ERP acts as the central database for financial and operational systems. In a distributed environment, these repositories are responsible for storing and providing data to the client machines. Usually, such systems are hosted in high end powerful systems or mainframes and expose web services for handling requests – both query and updates from the devices. The functionalities of the host layer can be summarized as follows:

(a) Centralized data storage and management: The host server system acts as a central repository for storing and managing bulk amount of data. This centralized approach ensures that data is consistent, easily accessible, and can be efficiently managed and updated in a collaborative environment. It simplifies and abstracts the complexities of data management as well.

(b) Security and access control: Hosting data storage on a server enables to implement robust security measures such as access control, authentication, and authorization mechanisms at the server level. This ensures that the sensitive data is protected and users or client applications can only access information for which they have the appropriate access rights and permissions. Also, it is easier to monitor and manage security measures implemented in such server systems.

(c) Scalability and performance optimization: The host server system can be optimized for performance and scalability through concurrent operations and efficient workload distribution. Load balancing techniques can be implemented to ensure that the system can handle increased traffic and user demands, making it more scalable and responsive.

The middleware layer can be hosted either in the host layer or in a dedicated system depending on the application and other security needs. While dealing with sensitive data, the middleware is preferred to be hosted in inhouse servers. A detailed discussion on the client agent and the middleware component and their interactions is provided in Chapters 5 and 6, respectively.

## 4.3 Characteristics of the Proposed Model

The proposed transaction model is designed to transmit data in a wireless mobile environment that faces many inherent issues related to connectivity, security, performance and resource scarcity. Wide spread use of mobile Internet facilities has transformed the way people interact, communicate, access information, and use digital services. This demands a better service quality and user experience in the field of communication. The major characteristics of the proposed model are [148]:

(a) Compactness: The proposed model makes use of the lightweight LXML as the format for data interchange. The LXML format is expected to be considered as an alternative to conventional data interchange formats due to its concise nature and other relevant performance advantages as discussed in the Chapter 3.

(b) Paginated: The amount of data communicated between the enterprise servers and device may vary with applications and users. Both the communication protocols and the device characteristics impose size restrictions on the amount of data communicated in a packet. A data transmission beyond the actual limit should be split into multiple transactions at the source and this process is known as pagination. Such transactions are rejoined later before parsing and extraction at the destination.

(c) Reliability: In an enterprise business model that has users with mobile hand-held devices on the client side, data delivery is very crucial due to the following factors: The mobile devices may join and disconnect from the network at any time. It may use different options for connectivity ranging from short range communication to Wi-Fi. It is to be noted that even when the communication is based on TCP / IP protocols, there are many issues such as addressing mobility issues due to inefficiency of network layer protocol and inefficiency of application and transport layers [166][167]. Hence mechanisms to ensure reliable data transmission are to be done at the application layer. In the proposed model, a handshaking mechanism is used to ensure the reliability of the data transfer. After transmitting a set of packets, it waits for its acknowledgement. Acknowledgement contains sequence numbers that are successfully delivered or not delivered properly. It can be either a ACK (all packets in the set are properly received) or NAK (atleast one packet in the set is not received). A negative or no acknowledgement results in retransmission of the specific packets.

(d) Disconnectedness: One of the significant advantages of mobile computing is data availability and the support for anywhere computing. The mobile client device can access data from anywhere having the coverage. Also client devices may join or leave the connectivity at any time. The connectivity of a device can be lost due to many reasons such as losing proximity from a Wi-Fi router or node, power failures, lack of battery backups, and other kinds of hindrances. The data availability is a crucial factor in a multi - user distributed environment. Relying on database replication techniques is not advised due to resource constraints in the handheld device [168]. To address the issue due to the "disconnectedness", a message queuing system called disc_queue and a poll service is included in the proposed model. Messages transmitted during disconnected state will be kept in the queue. Poll service is responsible for checking and updating the device network status. Once the network connectivity is restored, the transactions pending in the queue will be handled.

(e) Distributed: The proposed model supports and handles distributed processing in the sense that the client component resides in the handheld device and the middleware component is installed in another powerful machine. The middleware software can be resident on the in-house server or a dedicated system. Data is stored in a central repository to which all queries and updates are directed to. This arrangement also offloads high computing tasks from the mobile handheld device to the servers, paving a way for light load on the mobile handheld device.

## 4.4 Summary

The three layered architecture for the proposed Lightweight, Reliable, Paginated, Distributed, and Disconnected Transaction Model is designed to provide a robust and flexible framework for managing transactions in a distributed and disconnected environment. This architecture is particularly well suited for applications that require reliable data exchange and transactional support across multiple heterogeneous nodes or devices. The key components and layers of this architecture include the client agent, the middleware, and the host server. The model offers a comprehensive framework for managing transactions in complex and dynamic environments. It balances the need for lightweight operations with the critical requirements of data reliability and consistency in distributed and disconnected scenarios, making it a suitable choice for a wide range of applications in distributed environments.

Chapter 5

# Design and Implementation of the Client Agent Layer

## 5.1 Introduction

This chapter discusses in detail about the design and implementation of the first layer of the proposed transaction model - the client agent layer.

## 5.2 The Client Agent

The client side of the proposed transaction model is resident on the handheld device and coordinates the activities in the handheld device. This is the section where the user directly interacts with the system. The client side interacts with the gateway interface for the proper communication of user data. The functional block diagram of the client system is shown in Figure 5.1.

In the proposed system, the client side handles three types of message transactions as part of the communication between the device and the host server:

(a) Device initiated server updates: Applications use these types of message transactions to insert, update or delete the data to the host server. These types of messages originate from the handheld device and the size of such messages depends on the amount of data to be updated. Server updates are intended

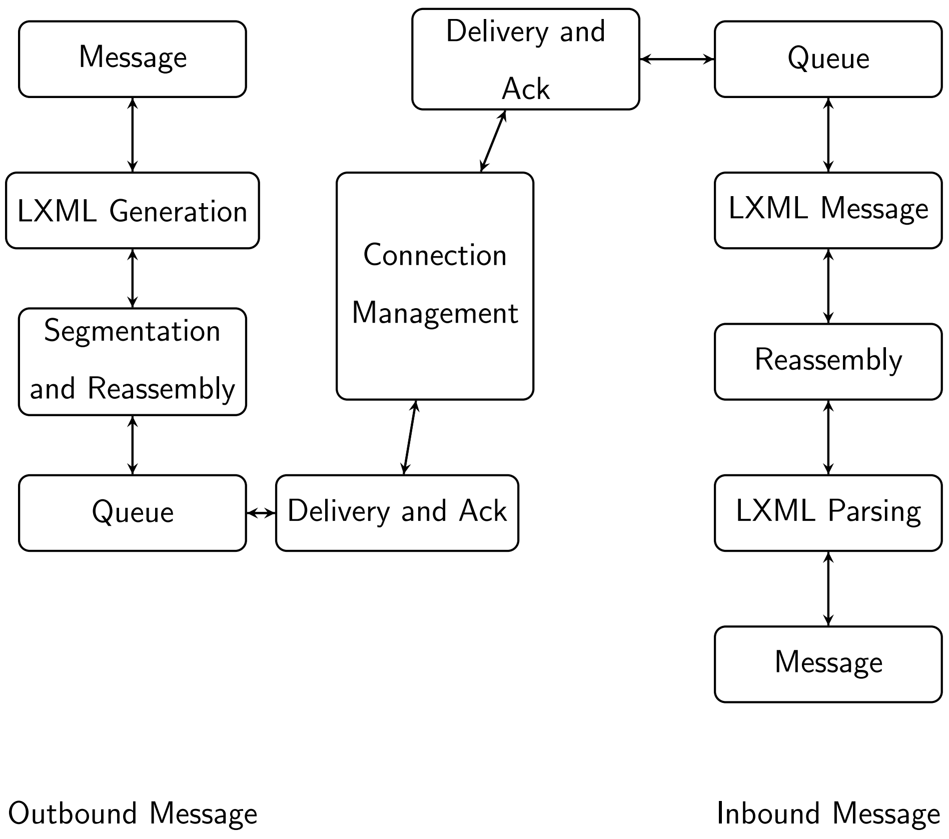

Figure 5.1: Functional block diagram of the client agent

to upload the device changes to the server so that other devices and users get the latest updates. Such server updates are important to maintain data consistency and integrity. These types of transactions are identified using a character ‘U’ in the message type field in the header.

(b) Device initiated request / response messages: This transaction is used to query and fetch data from the server to the device. The client device requests the desired data specifying its filter conditions. Server, in turn, responds by providing the required data matching to the given filter. The amount of data fetched depends on the filter condition and the application type. These types of message transactions are used to update the devices with latest data from the server upon the request from the user of the device. A character ‘Q’ in message type is used to identify the request transaction and ‘R’ is used to identify response transaction in the message header.

(c) Host initiated device updates: These types of messages originate from the gateway / server without the request from the device. Gateway / server intentionally pushes such data to the device. The data in the host server may change due to some events in the server or due to the updates made by other devices in a collaborated multi - client environment. In such a collaborative environment, the latest updates should be pushed to all collaborating devices to maintain consistency. Ignoring such updates leads to data inconsistency. To identify such host initiated push messages, a message type ‘H’ is used in the header.

The client agent has a different set of roles while dealing with messages received from the gateway interface and the device generated server intented messages. The major activities carried out in the client side are (i) connection management, (ii) message transmission, (iii) message expiration management, (iv) message retransmission, (v) disconnectedness, (vi) pagination (segmentation and reassembly), (vii) delivery and acknowledgement, and (viii) message parsing and processing.

### 5.2.1 Connection Management

A connection should be established between the client and middleware system before transmitting messages. Whenever the application is running in the handheld device, it will attempt to establish and maintain a constant connection with the middleware interface. Various communication protocols such as HTTP, TCP/IP, WCTP or any mobile device specific protocols are supported by the middleware interface. The Android system uses HTTP protocol for connection establishment. When a connection is established, the first packet will be sent as a dummy packet. The dummy packet will always be sent from the device to the middleware. This dummy packet will serve as a medium for device registration. Usually dummy packets will not be acknowledged by the middleware. The interface will never send a registration packet to the device. The registration packet contains following information:

- Packet Type: A packet type “dev-reg” is used to indicate that the incoming packet is a registration packet
- Packet Length: The size of the packet
- Device Name/Address: Name of the handheld device. This name is used for transmitting messages from the interface.
- Name/Value Pairs: Additional information if any, in the form of name value pairs.

A sample device registration request is as shown below.

```
TYPE DEV-REG
ADDR  <DEVICE ADDRESS>
NET  IPH
STATUS  DEV ACTIVE
Len : 4
```

Once the connection is successfully established the messages will be transmitted and the interface can properly trace out whether the data is successfully delivered or not. A connection can be disrupted due to many reasons such as network failure, and coverage issues. Once such issues are resolved, the system tries to re-establish connection. If the connection cannot be established or fails, instead of discarding the message and forcing the users for a repeat attempt, arrangements are made to preserve the message and to resend it automatically when the connection is regained by the client device.

### 5.2.2 Message Transmission

The client agent residing in the handheld device is capable of handling different types of messages generated in the device or received from the host system. On the device side, the application generates text messages and the text messages are converted to LXML format for transmission to the middleware. This LXML message is generated in accordance with the LXML Schema for that transaction. A sample device generated request message for data synch is as shown below:

```
String msg = reqTransId + "|1" + userId + "|2" + pwd +
"|2" + transactionName +"|2" + "" + "|2" + "data";
```

Business applications send a text message in the above format to the messaging API for transmission. The API handles serialization, segmentation and transmission of the LXML messages. Larger LXML messages are fragmented into multiple segments depending on the size of incoming message (pagination). Each segment will have a part of the actual message to be transported. The size of the segment can be configured in the device depending on the client device characteristics. The default size is set to 1400 bytes, the typical Internet packet size which avoids packet fragmentation downstream [169]. An LXML message with multiple segments will be sequentially numbered and transported in a serial manner after attaching the header information. A typical packet format is as shown below.

```
Packet  [<Header><Segment>]
```

The segment header consists of the following details:

(a) Message type: A single character message type field is used to identify the message type. A character "U" indicates that the message is used for updating the host server that are initiated from the device, "Q" indicates that the message is a query request generated from device to the server so as to make the device consistent with the server, character "R" indicates that the response message generated from the server, and "H" indicates that the message is an unsolicited message that is pushed by the server onto the device without any request from device side.

(b) Sequence number: Every segment is sequentially numbered for identification. A hexadecimal numbering system is used as sequence numbers. This unique reference number associated with each segment helps in retransmission of lost or damaged segments. The client must maintain an association between the sequence number and the message content. The sequence number of the request transaction is copied to all the response messages corresponding to that particular request. This helps in message / transaction tracking activities, debugging and clearing the message queues.

(c) Total number of segments: This field indicates the total number of segments generated for a particular user message. This count is obtained before transmission and attached to each segment. This helps to identify the last segment and end of a particular transaction.

(d) Device identifier: In a collaborative working environment, there will be multiple devices attached to a single middleware system and communicating with the same host. So there should be a way to identify the device. A device identifier is used to uniquely identify the client device. Device address is normally used for identifying the device and is configured in the middleware system. Middleware uses this identifier to correctly dispatch response messages to devices. This field is different from the destination address. Device initiated server updates and requests use middleware IP address as destination address.

(e) Checksum: A CRC checksum is calculated for the message and inserted in the header in order to identify any corruption or noise that is added intentionally or unintentionally to the message during its transmission. Upon receiving the packet, the CRC of each segment is recalculated and compared with the value available in the header to identify the corruption. Any mismatch leads to the rejection of the corresponding segment and needs retransmission.

(f) Acknowledgement: Acknowledgement is an optional piece of information attached in the packet to inform the counterpart whether the previous packet is properly received or not. This field contains the reference number of the previous packet successfully received on the other end. This option provides a piggyback mechanism rather than sending a separate message for acknowledgement. This field may not always be utilized, but plays an important role in server updates to maintain consistency.

A sample server update message is as shown below.

```
U01010021A0000020220531T12:18:13+05:30
DRET<01>adminuser<01>Test<01>EMP-DETAILS
```

Decoding of the above host update message is as shown in Table 5.1.

Header of a HTTP request and response received is as displayed in Table 5.2.

After a transmitting a packet, the sender waits for another transmission until the first of the following events occurs:

1. An acknowledgement is received on the device
2. The connection is lost or disturbed
3. A timeout period of 60 seconds elapses without an acknowledgement for the previous packet

### 5.2.3 Message Expiration Management

There is an expiry for every message transmitted over the network. Message expiration occurs when all the individual segments of the message expire. Expired

Table 5.1: Decoding a transaction

| **Component** | **Meaning** |
|---|---|
| U | Host update message |
| 01 | Message segment number |
| 01 | Total number of segments |
| 0 | Host number |
| 02 | Length of device generated message |
| 1A | Device generated reference number |
| 00 | Check sum |
| 00 | Reference number acknowledged (piggyback) |
| 20220531T12:18:13+05:30 | Date and time |
| DRET | Transaction name followed by data fields |

Table 5.2: Header of a request and response received using HTTP

| | |
|---|---|
| POST <**DEVICE ADDRESS**> | HTTP/1.1 200 OK |
| HTTP/1.1 | Content-Type: text/plain |
| Content-Type:application /x-www-form-urlencoded | Date: Fri, 20 May 2022 11:12:23 GMT |
| Connection:close | Content-Length:256 |
| Via: ANDR-GTWAY..Accept:*/* | < **MESSAGE**> |
| Host:127.0.0.1:8080 | |
| Content-Length:1024 | |
| Chksum:AA | |
| SEQNO:01 | |
| <**MESSAGE**> | |

messages automatically generate a negative acknowledgement to the client application. Expiration occurs in three scenarios: (i) acknowledgement timeouts, (ii) transmission timeouts, and (iii) message download timeouts.

(i) Acknowledgement timeout: It is the amount of time the client system waits for an acknowledgement. Once this pre-set interval elapses, the message is considered as expired and the system attempts for retransmission.

(ii) Transmission timeout: It is the number of attempts the client device should attempt for message transmission when the first attempt fails. The transmission timeout occurs mainly due to connectivity issues and are informed to users via some alerts.

(iii) Message download timeout: This timeout occurs while downloading messages to the client system. It is the amount of time the client should wait for message segments. Message download timeout occurs when at least one segment in a multi segment message fails to download.

### 5.2.4 Message Retransmission

On the client side, a segment or a message should be retransmitted if it does not receive any acknowledgement for a pre-set timeout interval or the message segment expires. When a message expires, all the constituent segments are considered as expired. Status of such expired messages will be updated in the message queue. When a message expires, the entire message should be retransmitted.

### 5.2.5 Measures to Ensure Disconnectedness

The most important characteristic of a distributed wireless environment is relocation (mobility). The device may move from one point to another point according to its mobility pattern. During this movement the chances for disconnection is higher. Disconnectedness in a mobile environment refers to the state where the device loses the network connectivity and is unable to send and receive data or signals. This disconnection may be temporary or persistent according to the situation. A mobile device can be disconnected due to many reasons such as:

(a) Physical obstruction: Network signals can be obstructed by many physical structures like buildings. This can cause disconnectedness or reduced signal strength.

(b) Lack of coverage: There are many locations especially remote and rural areas which are out of coverage. When the device enters such locations the device experiences disconnectedness.

(c) Congestion: During peak hours, the network may experience congestion. This congestion may lead to dropping of packets while handling large volumes of data.

(d) Device or SIM card issue: Reasons such as device hardware or software malfunctions, outdated operating system, firmware and other software, and issues with the SIM card may cause interruption in the network.

(e) Operator related issue: Technical or infrastructure related issues from the operator side can lead to outages and disconnection.

(f) Lack of plan or validity: Lack of appropriate plan and validity can lead to disconnection.

(g) Roaming related issues: Device that moves from the home network may experience disconnection due to reasons such as inappropriate roaming plan, incompatible technologies, and lack of roaming agreement with the operator.

A disconnection will adversely affect the ongoing transaction. To address this issue the following facilities are proposed in the client: (i) a message queue called disc_queue, and (ii) poll service.

#### Disc_Queue

The client agent residing in the mobile device and gateway interface maintains a buffer to hold the messages that are failed to transmit. This buffer system maintained internally is called disc_queue. When the device gets disconnected, the messages are pushed to this internal queue. When the user attempts to send a message in a disconnected state, the message is temporarily stored in the queue and the user gets notified. Once the connectivity is regained, the messages are transmitted to the intended recipients automatically without user intervention and queues will be cleared.

### Poll Service

The poll service is a component responsible for monitoring the network status in the device and updating other components in the system. The poll service gets activated when the device goes offline. This thread is periodically run to check the device state. When the device regains connectivity, it sends a poll packet to the middleware for updating its current status. Later the message queue is checked for messages that are not transmitted due to disconnection. The transmission unit attempts to transmit the queued messages one by one. The activities performed by the poll service is outlined in Algorithm 2.

The poll interval can be configured according to the application need. Configuring a shorter poll interval makes the device busy and consumes more CPU time whereas a wide polling interval results in a huge lag in transmission which is not advisable in critical applications. So it is recommended to choose the polling interval as a trade - off between performance and delay. If the interval is not configured, the system automatically chooses the gap in an exponential manner. A sample poll request is as given in below.

```
POST /<DEVICE ADDRESS> HTTP/1.1
Content-Type   : application /x-www-form-urlencoded
Connection     : close
Via            : ANDR-GTWAY..Accept:*/*
Host           : 127.0.0.1:8080
Content-Length : 0
<DEVICE-INFO>
```

The system makes use of Android system calls to implement the poll service.

**Algorithm 2:** Polling mechanism

```
Data: Polling intervel as set by the user and device addresses
Result: Connectivity status of each device
When the device wakes up or regains connectivity, it sends a poll packet
  to the middleware component
The middleware identifies the device, updates the status of the device and
  checks the queue for any messages that are not due to the unavailability
  of the device
if there is atleast one message in the queue then
    if the previous segment is the last segment then
        complete the transaction
    end
    if acknowledgement is received then
        send the next segment from the queue
    end
    else if a poll request is received then
        resend the previous segment
    end
    else
        send the message (packets) from the queue
    end
    Upon receiving the segment successfully in the device, the device
      processes the segment and sends an acknowledgement back to the
      middleware. Otherwise, the poll request is repeated.
end
```

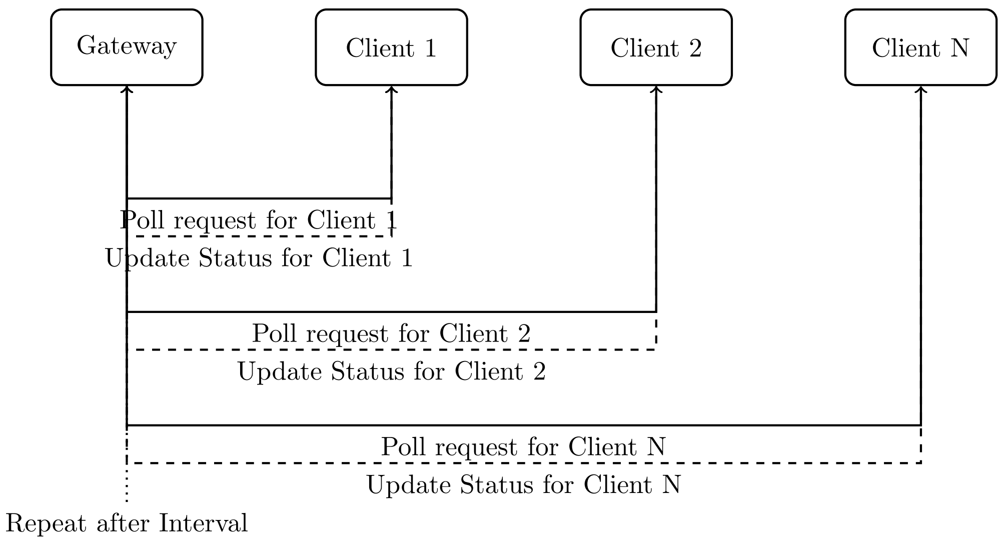
Gateway
Client 1
Client 2
Client N
Poll request for Client 1
Update Status for Client 1
Poll request for Client 2
Update Status for Client 2
Poll request for Client N
Update Status for Client N
Repeat after Interval

Figure 5.2: Protocol diagram for polling service

The time required for polling related activities is negligible and depends on the device capacity as well. Hence the polling service does not incur additional overhead to the client and the gateway interface. The entire polling mechanism can be represented using the protocol diagram given in Figure 5.2.

### 5.2.6 Pagination (Segmentation / Reassembly)

There is always a limit for the amount of data being transmitted between devices and the middleware in a single stretch. The limit depends on the protocol and the network carrier used at the time of transmission. For the smooth handling of messages, the limit is set to 1400 bytes in this model.

Pagination is referred to as the mechanism to divide a large message into chunks called segments, so that it is easier to transmit and process in the device and middleware. When the size of the message to be transmitted is higher than the limit, pagination of the message is carried out. So a large message reaches the receiver end in chunks. These message chunks should be rejoined before parsing and processing. This rejoining of message chunks to original message is termed as reassembly. Pagination and reassembly are required on both ends (device and middleware) to support transmission of large messages.

Reassembly of segments in handheld devices occurs when the device receives a response message or unsolicited push from the server through the middleware. Segment reassembly is carried out after all the segments are received in proper manner with correct checksum. After reassembly the LXML data will be parsed, stored in the device data store, and the message queues are cleared. In middleware, reassembly of messages is carried out after the device uploads bulk data to the server.

Every message segment has a unique sequence number to support the rejoining at the receiver side. A two byte hexadecimal sequence numbering is implemented in the proposed model and is kept in the header. Segments are rejoined in the order of sequence number and it is very easy to identify missed or lost segments. There can be situations where segments are partially received or few segments are not at all received. This situation is identified upon the receipt of an unrelated segment. To address such situations, the last segment of a message is always tracked. Upon the receipt of the last segment if there are still a few more segments pending to be received, a request is sent to retransmit the lost segments. If the missing segments are not received even after the retransmission request then the entire message is discarded.

Pagination is a mechanism to handle bulk messages. In addition to this, it helps in retransmission of lost messages. When a message is partially lost during transmission, only the lost segments need to be retransmitted instead of the entire message. This arrangement reduces the bandwidth and transmission costs to a great extent, especially in unreliable and untrustworthy wireless environments.

On a receiving a segment, the activities that are performed by the receiver can be summarized as in Algorithm 3.

If a recipient receives the message partially (in the sense that few segments arrived and few segments lost during transmission), followed by a set of unrelated segments, the recipient should discard the partially received segments. Segment acknowledgement mechanism can be expressed as shown in the Figure 5.3.

**Algorithm 3:** Segment acknowledgement mechanism

```
Data: A segment received in the device
Result: A positive (ACK) or negative (NAK) acknowledgement
The recipient of a segment calculates the checksum of the segment
Receiver calculates the header checksum of the data packet
if the header checksum of the segment matches with the calculated
  checksum then
    Extract the sequence number and verify it is not duplicate
    Ignore the duplicate segments
    Forward the segment to the upper-level for processing
end
else
    Ignore the newly received segment
    Request for retransmission
end
if the received segment is the last segment then
    Send back an (positive) acknowledgement
end
```

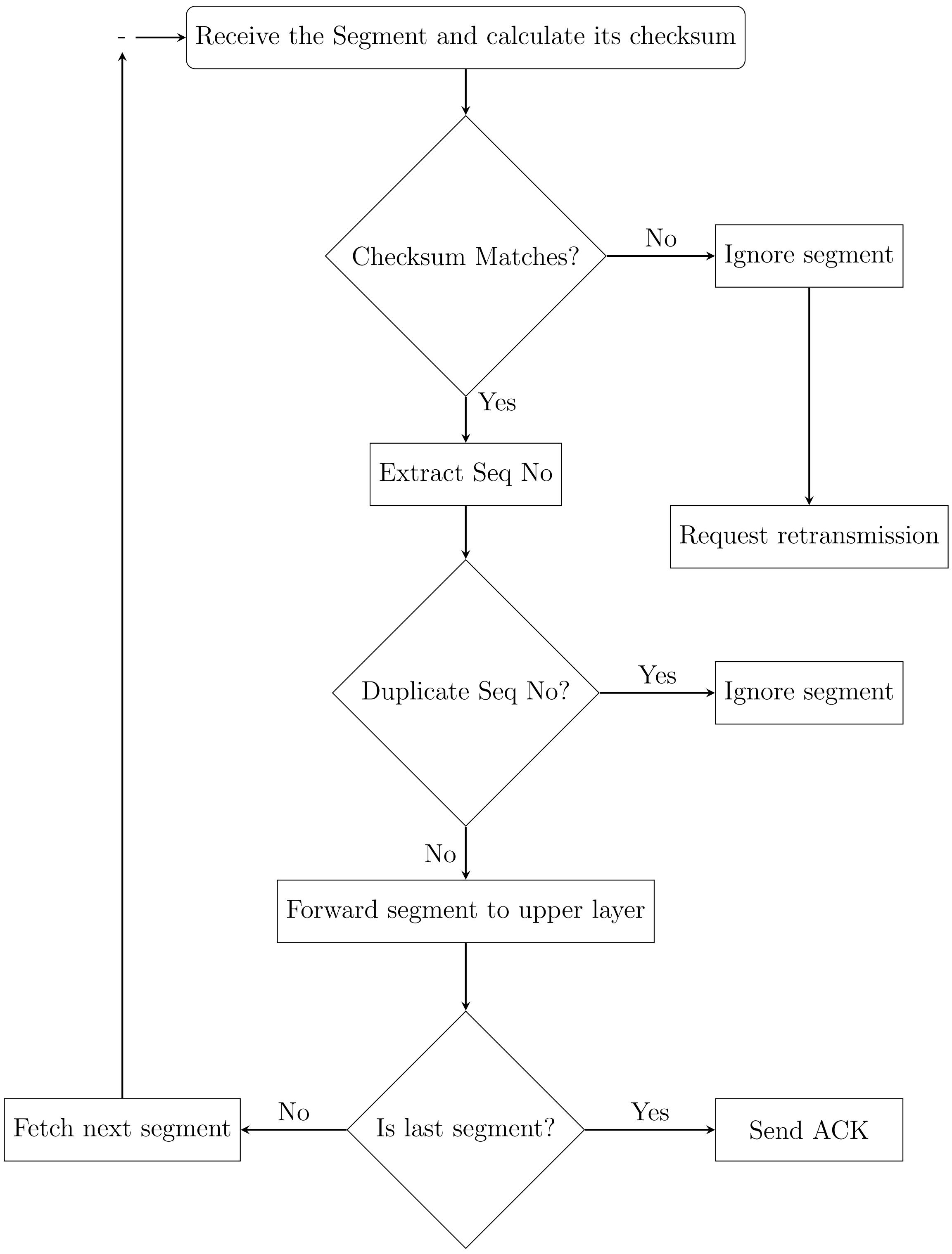

Figure 5.3: Protocol diagram for segment acknowledgement mechanism

### 5.2.7 Delivery and Acknowledgement (Mechanisms to Ensure Reliability)

A reliable and successful communication between the device and the middleware should ensure that the message is reached in the recipient in proper format. Sometimes the protocol used for data communication such as HTTP is not reliable and does not guarantee the message is correctly received. A reliable data transmission is very critical in case of sensitive or business data; any failures related to reliability can adversely affect the overall reliability of the system. Acknowledgement mechanism is used to ensure the message segments are properly received on the other end. Hence this mechanism is important to ensure the reliability of data transmission, even in the absence of reliable transport protocol mechanisms.

If the segment transmitted from the sender is received at the destination end, the first step is to verify whether the message chunk is received without any loss or corruption. CRC checksum method is used to detect errors that may occur during transmission. The sender computes the checksum for a segment and is attached in the header. At the receiver, the CRC checksum is recalculated and any mismatch in checksum indicates an error or corruption during transmission. In such cases, a negative acknowledgement (NAK) with corresponding segment number is sent to the source for resending the particular segment. In case multiple segments fail in CRC checks, a single negative acknowledgement is generated having all segment numbers that require retransmission. If the checksum matches, then a positive acknowledgement (ACK) is sent to the source; subsequently the source can also clear the message chunk from the queue. There can be scenarios where the acknowledgement (either ACK or NAK) itself can be lost during transmission. In such cases the source never receives any updates and waits for a predefined timeout interval. When the timeout occurs the segment will be resend.

The acknowledgement can be attached along with the outgoing segment sent back to the sender. This method reduces the network overhead leading to less network congestion and improved efficiency.

The delivery and acknowledgement mechanisms implemented in the system (which is performed by the receiver) can be summarized in Algorithm 4.

**Algorithm 4:** Packet delivery and acknowledgement mechanism

```
Data: A packet received from the gateway/device
Result: Positive acknowledgement (ACK) or negative acknowledgement
        (NAK) with failed sequence numbers
Repeat the following steps, when the receiver gets a packet.
Calculate the CRC checksum of the incoming packet
if the calculated checksum is same as that of the stored CRC available for
 the packet then
    Send a ACK to the source
    Forward the entire data packet to the next level for processing
end
else
    Ignore the newly received faulty packet if previous segment is also
     failed then
        Add the sequence number to the list of failed sequence numbers
        if last segment received then
            Send a NAK to the source with all failed sequence numbers
        end
    end
    else
        Send a NAK to the source with sequence number
    end
end
```

The above given delivery mechanism can be represented using the protocol diagram given in Figure 5.4.

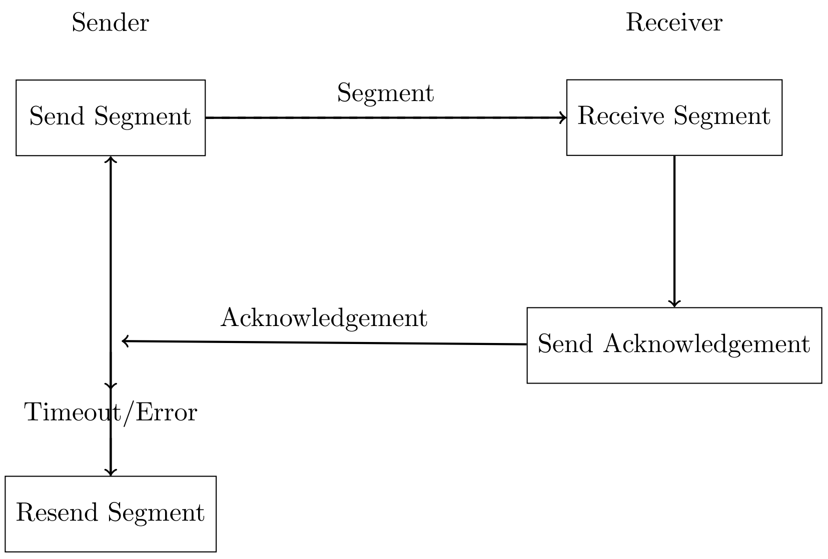

Figure 5.4: Protocol diagram representing packet delivery and acknowledgement mechanism

### 5.2.8 Message Parsing and Processing

The data communicated between the device and the middleware system is in LXML format. A message received either in the middleware or in the client device should be parsed before extracting the data. The LXML messages are parsed and validated using the LXML Schema. Any mismatch during the validation process generates negative acknowledgement and requires retransmission of the message.

On the device side, the valid LXML message can be extracted with the help of database objects (DAOs). The DAOs thus generated out of LXML messages can be loaded to the database directly. In the middleware side, the extracted data is converted to a format acceptable by the host server. A detailed discussion on LXML parsing and extraction is discussed in Chapter 3.

## 5.3 Implementation

The client side of the messaging model is implemented as a messaging API. This API is a lightweight collection of packages that supports reliable communication between Android mobile devices and the middleware interface. The messaging API is a highly scalable component that can be added as a plugin to applications that need to transfer messages. This API is based on HTTP/HTTPS protocol and provides support to segments. It consists of the following components / packages: (i) messaging service, (ii) poll service, (iii) database connection, (iv) listener and DAOs, (v) message queue, (vi) helper and utility classes, and (vii) configurations.

### 5.3.1 Messaging Service

Messaging service is responsible for handling the request and response transactions. This is implemented as an asynchronous service. For every request, a separate thread is created and is destroyed upon when the response is handled or the request is timed out. The messaging service obtains the configuration information from the configuration database table or from the config file. Only one instance of the messaging service class is created at a time.

This module makes use of two main classes: *LXMLMessage*, and *LXMLMessageHelper*. The *LXMLMessageHelper* class is responsible for activities such as pagination, sending requests, generating hexadecimal sequence numbers, formatting LXML messages, extracting message and status information from host response. This class provides methods for invoking a response handler for each host response. A sample LXML message created is as shown below.

```
String msg = reqTransId + "|1" + userId + "|2" + pwd + "|2" +
transactionName + "|2" + "" + "|2" + "data";
```

### 5.3.2 Poll Service

This service is responsible for handling the poll service described earlier. This is implemented as an asynchronous service. The polling interval is configurable by the end user and can be provided in the configuration file.

The polling service is handled by two classes: *PollerService* and *PollerHelper*. The *PollerService* class implements the *AsyncTask* threading service. This service initiates the polling operation; the operation is repeatedly executed until there are no messages with status as SENT. This includes all messages that failed to be sent due to network coverage problems. This class is also responsible for monitoring the connectivity status of the device.

*PollerHelper* class provides methods to invoke appropriate listeners once the response is received from the host. It also sends the LXML message to the host and receives all the chunks of responses back and creates a response message out of it. It generates and sends ACK messages when response chunks are received back from the host.

### 5.3.3 Database Connection

API is flexible enough to retain the configuration information and details of the messages transmitted from the client mobile device. A separate database is used to store LXML messages and configuration information, independent of the application database.

*MsgDBConnection* is the child class of *SQLiteOpenHelper* class that manages the database connection. This singleton class takes care that only one instance of the database object is active at a time to avoid data integrity issues. It provides the methods for inserting, retrieving records based on query criteria.

### 5.3.4 Listener and DAOs

The API provides a listener interface and a default listener. The application developer has to register his/her own listener and can implement this listener class as and when required. If no listener is associated with a particular response, the default listener is invoked. Application developer has to create DAO based on his/her database designs and it can be incorporated with the client.

There will be a separate listener interface for each response to be handled in the device end. A listener class should always implement *ILXMLResponseListener* interface. The default *LXMLResponseListener* class handles the response messages received from the gateway. This method will be invoked by the messaging API once it receives the LXML message from gateway.

*LXMLMessageDAO* is the database object that represents the LXML message. This class helps to readily store and update the LXML messages. It also sets the message status so that messages can be easily filtered out.

### 5.3.5 Message Queue

The messaging API is based on the store and forward concept. The API checks for network connectivity before sending a message to host. If no connectivity is available, the API stores the message in its internal database, updates the status and later when the mobile regains the network coverage, the message is transmitted. Application developers can conveniently configure the message timeouts.

This queuing service is implemented using two main classes: *ConnectivityService*, and *MessageQueueService*. The *ConnectivityService* class checks the availability of network coverage and returns the status. The *MessageQueueService* class is a child class of Android *Service* class and overrides the *onBind()* method. This service implements *ConnectivityListener* interface to listen to the changes in the network and retry pending messages that are yet to be sent.

### 5.3.6 Helper and Utility Classes

These are a set of commonly used methods such as creating reference numbers, formatting dates, string manipulations, defining messaging and database constants. The *HttpHelper* class available in the utility package is responsible for dealing with the HTTP protocol oriented request and response. It creates *LXMLMessage* objects compatible for TCP/IP networks. The *HttpHelper* class provides a static member to read the response data from the input stream.

Other classes available in this package include classes for manipulating timers, formatting dates, handling time zones and related activities.

### 5.3.7 Configurations

Configuration information consists of server address, device name, device address, server port, and polling intervals. Configuration is stored in the internal database and this is read prior to the loading of information from the configuration file. Initially the configuration is loaded from the file to the database by the messaging service.

*MessageConfig* is a serialized bean object for storing configuration details such as user id, server address, server port, device name, device address, host no, and polling interval. *LXMLMessage* is another serialized bean object that represents the LXML object. It store the LXML message id, transaction name, userid, device address, protocol, sequence number, message type, header type, host number, host id, server address, reference no, reference length, status, message content, message header, message, direction, created date, modified date, and message chunks.

## 5.4 Summary

The client agent layer of the architecture for the proposed Lightweight, Reliable, Paginated, Distributed, and Disconnected Transaction Model plays a pivotal role in managing various aspects of communication between the user's device and the middleware system. A summary of the major activities and responsibilities carried out by the client side is presented in Table 5.3.

The client side of this architecture plays an important role in managing the communication between the handheld device and the middleware system, ensuring reliability, efficiency in message handling, and transmission. This comprehensive set of functionalities contributes to the overall robustness and effectiveness of the transaction model in distributed and disconnected environments. The performance evaluation of the client agent is discussed along with the middleware layer in Chapter 7.

Table 5.3: Summary of the functions performed by the client agent tier

| **Component** | **Roles** |
|---|---|
| Connection management | Deals with establishing, maintaining, and monitoring the connection with the middleware system or gateway interface |
| Message transmission | Deals the transmission of LXML messages from the user's device to the middleware system |
| Message expiration management | Deals with message expiry and timeouts |
| Message retransmission | Deals with retransmission of undelivered and unacknowledged messages |
| Disconnectedness | Deals with offline communications when the devices loses connectivity |
| Pagination | Deals with transmission of messages of large size |
| Delivery and acknowledgement | Deals with tracking the delivery status of messages and acknowledgements |
| Message parsing and processing | Deals with validating and extracting data |

Chapter 6

# Design and Implementation of the Middleware Layer

## 6.1 Introduction

This chapter outlines the design and implementation of the middleware layer of the proposed transaction model. Middleware refers to a software component or service placed between software systems or applications. It is an intermediary component that enables integration, communication and data transfer between heterogeneous components. Middleware plays an important role in providing interoperability and boosting the overall functionality of the system in a distributed environment. It clearly abstracts the inherent complexity in integrating different platforms or technologies.

## 6.2 Need of Middleware

Middleware enables seamless collaboration between different application components and allows them to work in a flexible and efficient manner. Following are the major functionalities and services provided by a middleware component.

1. Component integration: Middleware enables integration of diverse components facilitating the smooth interaction and data sharing between different appli-

cations. It supports and handles data transformation, data mapping and redirection between different data formats or standards depending on the environment.

2. Communication: Middleware enables seamless communication and data transfer between applications. It may use diverse techniques to support this interaction.

3. Computational offloading: Middleware shifts the processing and other computations to a location where it is actually needed. This leverages the processing in mobile devices to a great extent.

4. Load balancing: Load balancing is an important functionality of middleware systems especially in wireless mobile environments, where the client layer is resource constrained and comparatively less powerful. The middleware can shift the processing from client to more powerful component. This ensures better processing speed and scalability.

5. Transaction management and monitoring: Middleware supports distributed processing and transaction management ensuring the operation is carried out by maintaining data consistency. It can monitor the client devices and manage the availability, performance of client components.

6. Data caching: Middleware is the apt place for cache location to improve the responsiveness of applications by storing frequently used data. This ensures improved system performance and better user experience.

7. Security: Middleware components can be used to enforce security mechanisms such as authentication, authorization, and encryption to protect the data and support reliable communication.

There are different types of middleware discussed in the literature. The middleware in this model belongs to the category of service oriented middleware system as it supports distributed transaction management, asynchronous mode of communication and other monitoring services.

## 6.3 The Middleware Component

The middleware component is the second layer in the proposed architecture that acts as an intermediary between the client side and the host server. The major design goals of the middleware system are primarily to bridge the technological differences between the server and the mobile device. In addition to this, the middleware component reduces the processing load of the comparatively less powerful handheld devices.

The middleware component is usually installed in a powerful dedicated machine or on the host server. It contains mainly two subcomponents: (i) gateway interface, and (ii) measures to ensure disconnectedness.

### 6.3.1 Gateway Interface

The gateway interface is the component responsible for establishing the communication with handheld devices, accepting the packets from the client devices, acknowledging the packets and extracting the LXML according to the LXML Schema stored in the interface. It then communicates the data to the server according to the format required by the server. On the other side, the middleware accepts the data received from the server and converts to the LXML messages and transmits to the mobile device concerned. The functional block diagram of the middleware component is shown in Figure 6.1.

The main functionalities of each subcomponent of the gateway interface are discussed below.

#### Device and Connection Management

This component is responsible for managing the connectivity with the device and the server. Every client device should be registered with the gateway interface for smooth communication. Client devices use device addresses and other device details for their registrations. A sample registration request is given in Table 6.1. The gateway interface will communicate only with the registered device and this

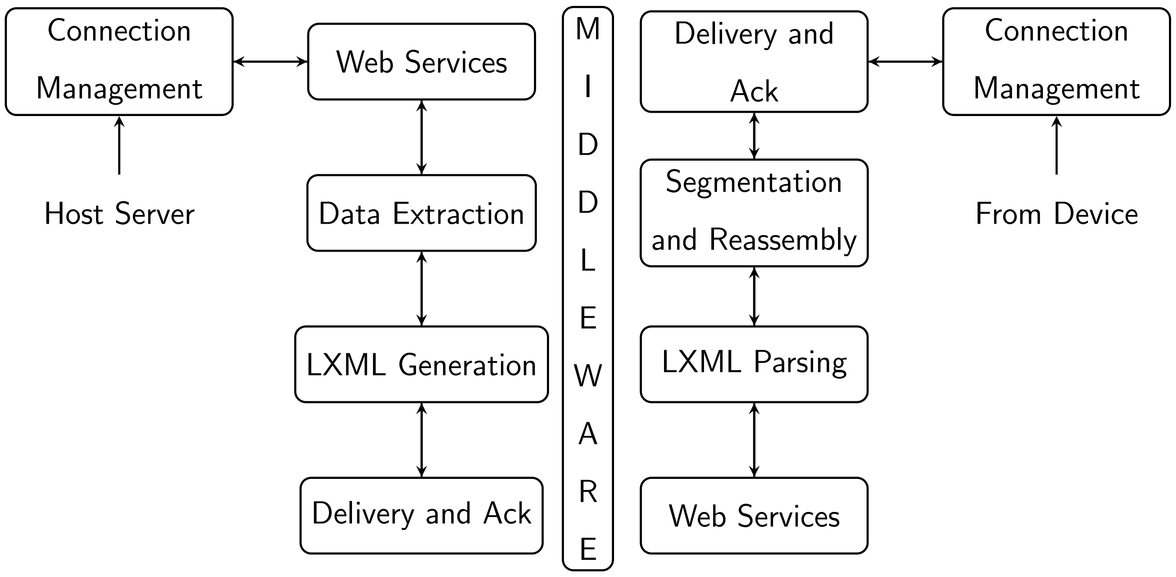

Figure 6.1: Functional block diagram of the middleware layer

serves as a security mechanism to ensure the authenticity of the devices used for communication.

Table 6.1: (a) Header of a device registration request, and (b) a sample status message of unauthenticated device

| | |
|---|---|
| TYPE DEV-REG | TYPE STAT-REQ |
| ADDR <DEVICE ADDRESS> | ADDR<DEVICE ADDRESS> |
| NET IPH | NET IPH |
| STATUS DEV ACTIVE | STATUS DEV INACTIVE |
| | TRANSIENT 192.168.0.1 |
| (a) | (b) |

Once the device is successfully registered, the gateway interface periodically monitors the connectivity status of each device and keeps a device registry for updating the status. Mobile devices can lose connectivity at any time and the devices may join or leave the network at any time. Once the device loses its connectivity, the gateway sends a poll packet to collect the connectivity status.

On the other side, this sub component also establishes connection with the host server.

### Message Handling

The gateway interface accepts data from the server. In most cases, the server exposes web services to the middleware for transferring data. Once the data is successfully received from the server, the middleware acknowledges the receipt.

The data available from the server is extracted and converted to LXML format before transmitting to the client device. When the size of LXML is far beyond the preset limit, the LXML message is split into separate packets. It attaches necessary headers to each packet and transmits to the designated client device.

When the client device is not available during the transmission, the gateway stores the message in the message queue and transfers the data once the device is available for communication. On successful receipt of the message at the client side, the queue is cleared. If the data is not reached the destination successfully, the gateway interface tries for the retransmission of the message.

When the gateway receives a LXML message from the client device, it parses the message. If the message is valid, it acknowledges the client. Later the LXML message is extracted using the LXML Schema and is converted to a format acceptable to the server and gets transmitted.

Sometimes the LXML message received from the client will be paginated - split into multiple transactions due to its bulkiness. In such scenarios, the individual packets are rejoined in the gateway before parsing.

Usually, a server – client communication involves many transactions. So, to handle each transaction there will be one specific function in the gateway interface. A detailed discussion on how the LXML messages are parsed and extracted is provided in Chapter 3.

### Delivery and Acknowledgement

As a measure for ensuring reliability in message transmission, the middleware acknowledges the messages received from both server and client devices. This is necessary in case an unreliable protocol is used for transmission.

The message transmitted from the client device will have checksum and encryption techniques for ensuring security. If the gateway receives the message, it verifies the header checksum. If the message is received in proper order, a positive acknowledgement is passed back to the device. A negative acknowledgement is given to the device if the message is somehow corrupted. Sending a negative acknowledgement can reduce the waiting time. To improve the efficiency, acknowledgement can be piggybacked with other responses to the devices.

### 6.3.2 Measures to Support Disconnectedness

The middleware component supports disconnected operations by using the message queue and polling service as mentioned in Chapter 5. In addition to these techniques, the middleware maintains device registry - a data structure to keep the status of each handheld device along with their device address.

#### Message Queue

Every LXML message generated in the middleware system for transmitting to the devices is stored in the message queue. Each transaction has one entry in the queue and contains the information such as LXML message, device address, sequence number, message type, date and time. LXML messages are stored as objects (*LXMLMessageDAO*). The messages stored in the queue are used for retransmissions in case of device disconnection or message corruption during transmission. The entries in the message queue are cleared immediately once the message is successfully delivered to the addressed device and corresponding positive acknowledgement is received in the middleware.

#### Polling Service

The middleware component periodically monitors the connectivity status of their collaborating devices by sending poll packets and updates the device registry. The polling interval can be configured at the gateway interface and is chosen as a trade - off between performance and delay.

## 6.4 Implementation

The middleware component is developed in Java. A web interface is used for tracking the participating devices of the messaging model are implemented in Java as an API and it is composed of following components/packages:

### 6.4.1 Device Registry

This service is responsible for handling the device registration request from the client. Once the device is successfully registered with this service, the component periodically checks the status of each device and updates the status accordingly.

### 6.4.2 Push Service

This service is responsible for handling the unsolicited push from the host. This is also implemented as an asynchronous service.

### 6.4.3 Request Response Handler and DAOs

The package provides a listener interface for handling the request from the device and will be directing these requests to the host server in the required format. It provides handlers for receiving responses back from the server, creates LXML messages and sends to the designated client device. There will be a specific listener for each request and response. If no listener is associated with a particular response, the default listener is invoked. Application developer has to create DAO based on their database designs and it can be incorporated.

### 6.4.4 Message Queue

The package is meant to store the messages sent to server or client devices. It checks for network connectivity before sending any message to the host. If no connectivity is available, the API stores the message in its internal database, updates the status and later when the mobile device regains the network coverage, the message is transmitted.

### 6.4.5 Helper and Utility Classes

These are a set of commonly used methods such as creating reference numbers, formatting dates, string handling, and defining constants.

## 6.5 Summary

The middleware component of the architecture plays a pivotal role in managing the communication between the mobile device and the host server. It acts as an intermediary layer that facilitates seamless communication and coordination between the client-side and the server-side of the system. The activities of the middleware layer is summarized in Table 6.2.

The middleware component abstracts the underlying complexities and makes it easier for the parties to communicate effectively and thereby ensures that transactions are managed seamlessly in a distributed and disconnected environment.

Table 6.2: Summary of the middleware layer activities

| **Component** | **Roles** |
|---|---|
| Connection management | Deals with establishing, maintaining, and monitoring the connection with the devices and the host (server) system |
| Message transmission | Acts as an intermediary for exchanging data between client and server |
| Message retransmission | Deals with retransmission of undelivered and unacknowledged messages |
| Disconnectedness | Deals with offline communications when the device loses connectivity |
| Pagination | Deals with the transmission of large messages |
| Delivery and acknowledgement | Deals with tracking the delivery status of messages and acknowledgements |
| Message parsing and processing | Receives LXML message from the client device and converts to a form acceptable by the server. Also accepts messages from the server and converts to LXML and transmits to the device concerned. Responsible for validating and extracting data before transmitting to the other end |

Chapter 7

# Performance Evaluation of the Proposed Transaction Model

## 7.1 Introduction

Performance evaluation is necessary to validate the effectiveness and reliability of a transaction model that is expected to be used in a wireless mobile environment that has many inherent limitations such as higher possibility of connection issues, lower bandwidth, and lesser battery power.

Performance evaluation with appropriate parameters helps to provide insights into the strengths and weaknesses of the transaction model. The criteria used to evaluate the performance of the proposed model includes: (i) turnaround time, (ii) content density, (iii) processing overhead in client and middleware, (iv) page wise transmission time, and (v) parsing overhead [153][154]. Performance evaluation and analysis based on verbosity of messages is extensively covered in Chapter 3.

## 7.2 Turnaround Time

It is defined as the time duration required for transmitting a message to the receiver and for receiving the response back to the sender. The turnaround time is an efficient measure of system efficiency. A shorter turnaround time results in faster response and is desirable for fast exchange of messages.

In business communication, messages are generated either in the handheld device or in the server. The messages thus generated are converted to LXML format before transmission. On the device side, the LXML messages are created before sending a request to the middleware. The messages received from the server are converted to LXML at the middleware. On receiving the LXML message, it should be parsed to extract the data contained within the message. Thus LXML creation and parsing is performed in both the client side and in the middleware. Parsing of LXML messages is performed when a response is handled on the client side. But in the middleware, LXML parsing occurs when the client sends a server request and LXML creation occurs when the host server sends data to the device.

In the proposed model, the turnaround time is calculated as the sum of the time taken for LXML message creation / parsing, transmission through the wireless / wired medium, middleware processing overhead and processing at the server. The performance evaluation regarding the creation and processing of the LXML messages is already discussed in Chapter 3.

The server side is usually hosted on a high end system such as CRM. Hence the processing time to perform request at the host server varies according to the host system. This processing cost is not considered for performance analysis.

Transmission time refers to the amount of time needed / required to transfer the messages from the device to the host server through the middleware. Since the middleware can be hosted in the server itself, the transmission time between the middleware and the host can be neglected. So transmission cost is the cost incurred during the transmission of messages between client and the middleware. Let P be the processing time of LXML messages, n1 is the size of request, n2 is the size of response, b1 is the upload bandwidth and b2 is the download bandwidth. The transmission cost is calculated using the formula 7.1 and 7.2, as shown below.

$$n1/b1 + P \tag{7.1}$$

$$n2/b2 + P \tag{7.2}$$

Adding 7.1 and 7.2, the total transmission cost is given by

$$C = n1/b1 + n2/b2 + 2 * P \tag{7.3}$$

A business transaction usually requires bulk amount of data to be transmitted. In such cases, data is communicated in pages, depending on the amount of data to be transferred. So the total transmission cost is calculated as the sum of transmission cost of each page. The transmission cost is calculated for a varying number of pages using the formula (7.3). For experiment purposes, a document having 1000 LXML objects is considered as a page. Each object contains up to 12 attributes [170]. The upload and download bandwidth is 4.8 Mbps and 5.1 Mbps, respectively. The results obtained are tabulated in Table 7.1.

Table 7.1: Processing time for uploads and downloads Vs page sizes

| **No. of Pages** | **Time Taken for Uploads (ms)** | **Time Taken for Downloads (ms)** | **Total Time (ms)** |
|---|---|---|---|
| 1 | 0.634 | 0.603 | 1.237 |
| 2 | 6.974 | 6.577 | 13.551 |
| 3 | 13.198 | 12.440 | 25.638 |
| 4 | 19.854 | 18.712 | 3.567 |
| 5 | 26.476 | 24.950 | 51.427 |
| 6 | 33.316 | 31.395 | 64.711 |
| 7 | 40.788 | 38.433 | 79.221 |
| 8 | 44.912 | 42.328 | 87.249 |
| 9 | 53.835 | 50.726 | 104.561 |
| 10 | 63.044 | 59.399 | 122.443 |

Table 8.1 shows the transmission time required for various numbers of pages for data upload and download. Figure 7.1 and 7.2 represents the time relationship for upload and download operations against varying sizes of pages, respectively. The combined result is presented in Figure 7.3.

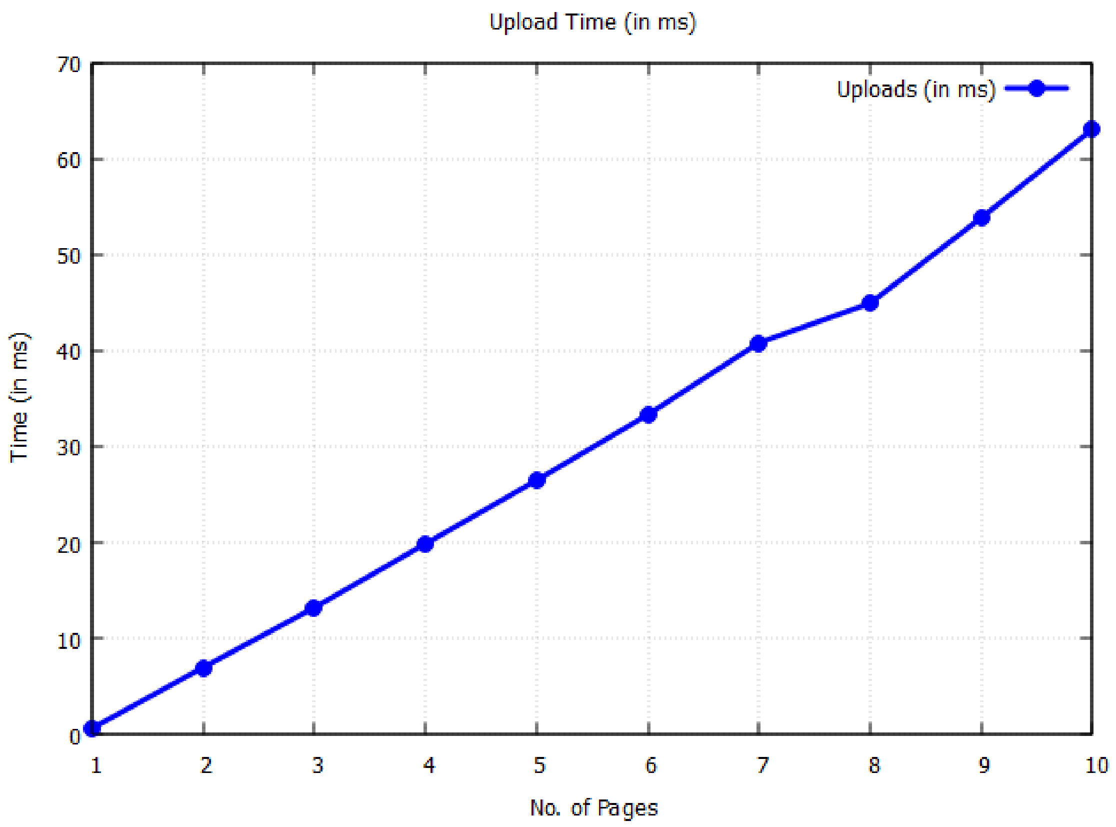

Figure 7.1: Upload time in ms (along Y axis) Vs the number of pages (along X axis)

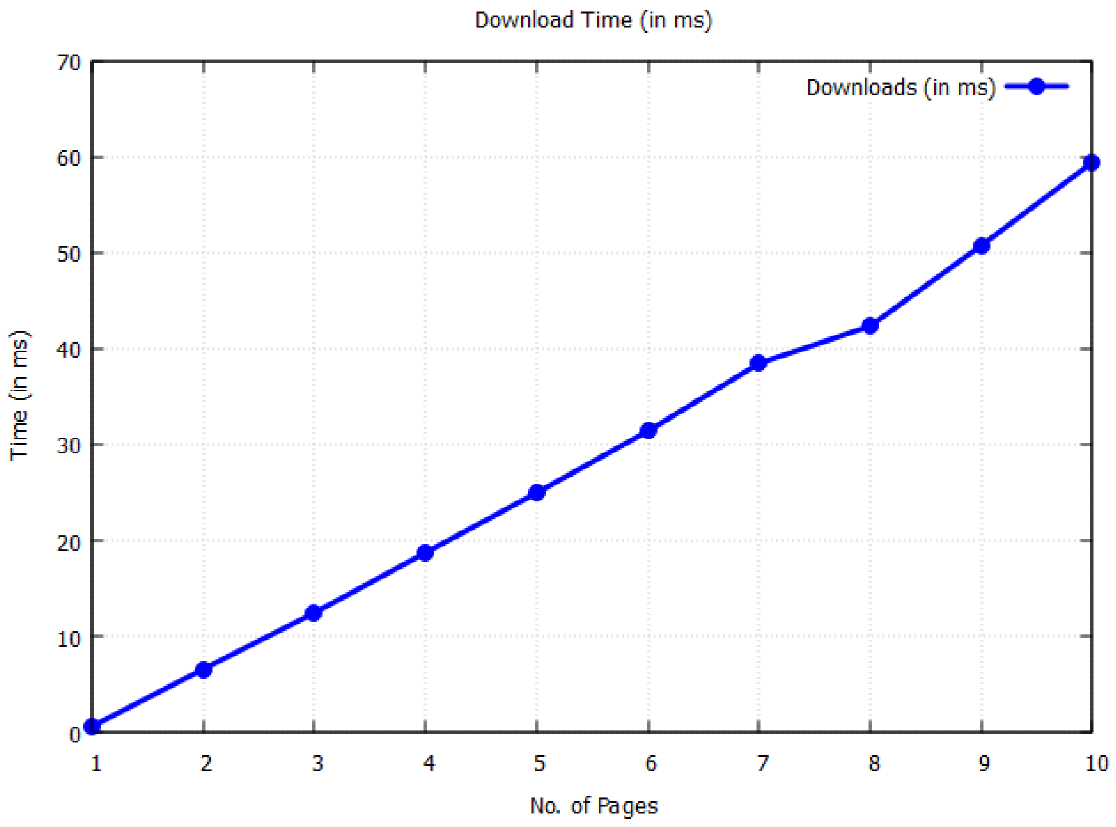

Figure 7.2: Download time in ms (along Y axis) Vs the number of pages (along X axis)

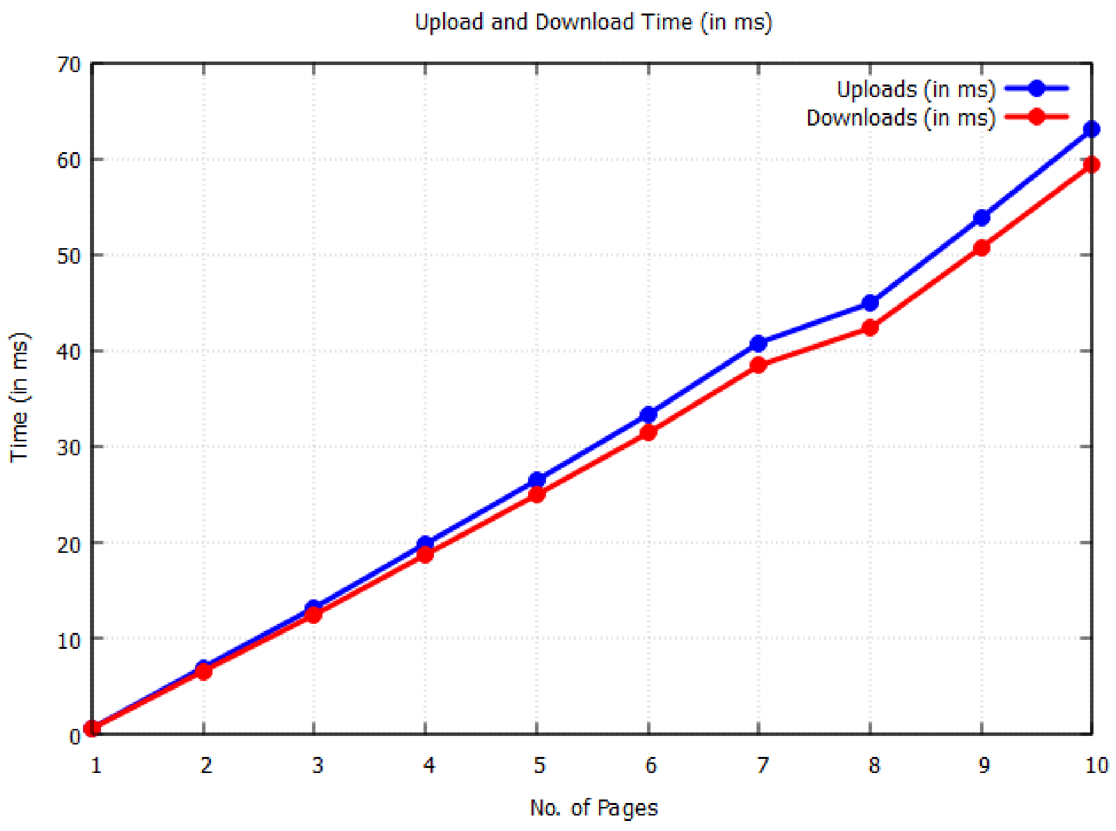

Figure 7.3: Upload and download time in ms (along Y axis) Vs the page size (given in X axis)

From Figures 7.1, 7.2 and 7.3, it can be inferred that when the document size increases, the document needs to be split to multiple pages prior to the transaction. So increase in document size makes transmission more expensive. In such a scenario, to increase the transmission efficiency, either bandwidth needs to be increased or the overall document size should be reduced.

The above result is calculated while transmitting data using the proposed lightweight data interchange format LXML discussed in Chapter 3. Verbosity advantages for LXML over XML and JSON reported in [148], are discussed in this Chapter. Since LXML is a compact format, it reduces the number of pages required, and hence the suitability of the transaction model based on LXML in terms of transmission time is also established.

## 7.3 Content Density

Improvement on content density for LXML over XML and JSON is already discussed in Chapter 3. The reduced message size and higher content density generously help to reduce the transmission requirements when the system has to send multiple packets in a stretch. This is a much needed requirement in enterprise CRM client - server transactions where bulk amounts of data is exchanged. This reduces the number of transaction iterations which in turn saves both bandwidth and transmission time. For the ease of evaluation, documents are classified into four categories depending on the size, based on the discussion in [65][171]. The average content density for various numbers of objects is shown in Table 7.2. Table 7.2 demonstrates a clear advantage for LXML, over XML and JSON.

## 7.4 Parsing Time in Client and Middleware

It is a matter of concern whether LXML creation and parsing in client and middleware add any additional overhead in the overall processing of message. The processing of LXML is a straightforward method using database objects (DAOs) for

Table 7.2: Content density comparison for XML, JSON and LXML

| **No. of Objects** | **XML** | **JSON** | **LXML** |
|---|---|---|---|
| 10 | 0.406 | 0.521 | 0.814 |
| 500 | 0.423 | 0.534 | 0.815 |
| 5000 | 0.337 | 0.438 | 0.767 |
| 10000 | 0.464 | 0.585 | 0.859 |

each repeating block in the response. The advantage of LXML in parsing time is already discussed in Chapter 3.

## 7.5 Transmission Time

To test advantages in transmission aspects, XML, JSON and LXML files of varying sizes [65][172] are considered. JSON and XML objects are transmitted by establishing connection using the *HttpURLConnection* class (Refer Chapter 3). LXML is tested using the new custom API developed.

In this experiment, the number of objects and the number of attributes in each object are kept the same for a given test. This experiment is conducted taking the packet size as 1400 bytes.

Table 7.3: Comparison of transmission time required (in ms) for XML, JSON and LXML

| **No. of Objects** | **XML (in s)** | **JSON (in s)** | **LXML (in s)** |
|---|---|---|---|
| 1000 | 2.231 | 1.251 | 0.634 |
| 2000 | 4.465 | 2.485 | 0 .634 |
| 3000 | 6.974 | 2.485 | 0.634 |
| 5000 | 12.197 | 5.031 | 1.276 |
| 10000 | 25.476 | 9.742 | 2.712 |

The transmission time taken for objects of varying size for XML, JSON and

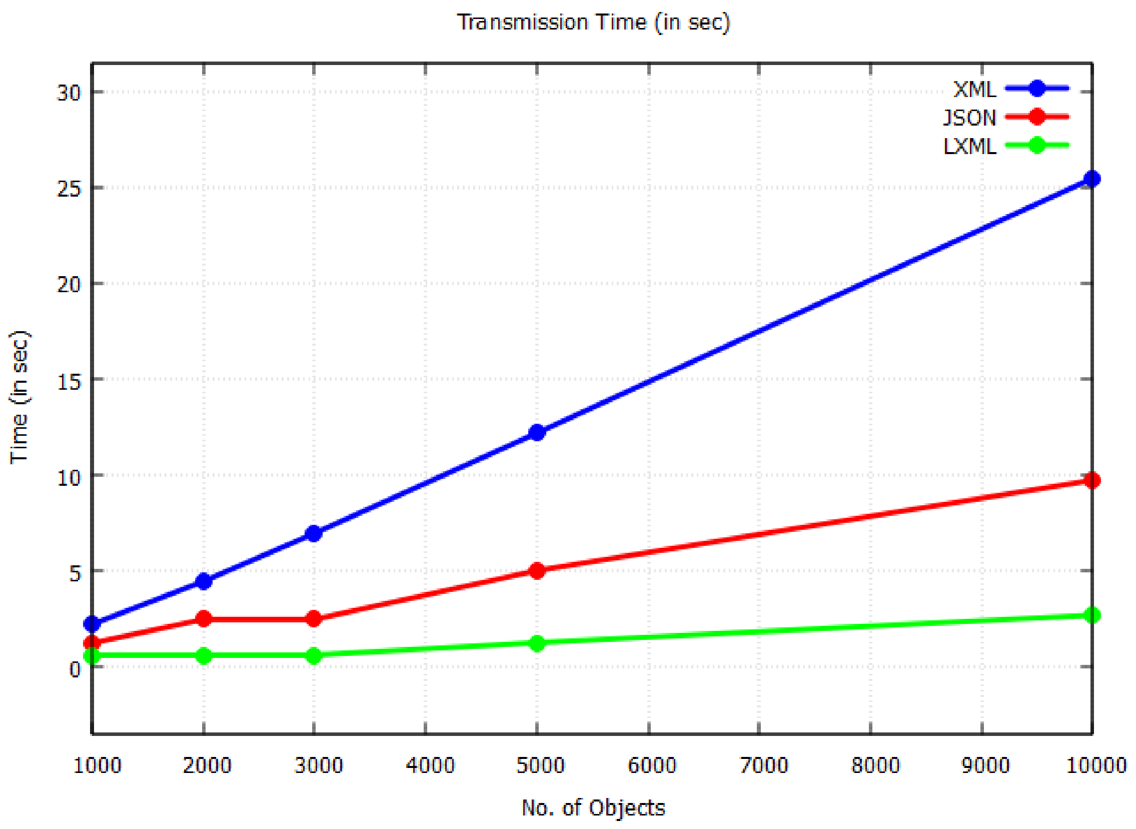

Figure 7.4: Comparison of transmission time (Y axis) for XML, JSON and LXML Vs the number of objects (X axis)

LXML is as shown in Table 7.3 and is graphically presented in Figure 7.4. It can be inferred that the LXML format takes the least transmission time among the three formats. When the number of objects increases, the time difference also increases.

XML and JSON based web services are the most commonly used approaches used for business communication where bulk amount of data is to be transferred. Since the verbosity of messages is having a direct impact on the size of the messages being transmitted, the LXML is the most suited format in such cases. It can be concluded that the LXML based transaction model has a clear advantage over other models in this regard.

## 7.6 Processing Overhead in Client and Middleware

Since LXML is directly created on the client side, there is no XML to LXML conversion on the client side while sending requests or server uploads. Similarly, LXML responses can be directly parsed and processed. So using LXML on the client side does not add any processing overhead in the client device. But there will be additional overhead for converting LXML to web service and back in the middleware system. The LXML messages should be converted to web services or to a format acceptable by the server during device to server communication. XML to LXML conversion is needed in server to device data transmission. Since this processing is done in the middleware system which is installed in the server, the processing power and resources needed is offloaded to the middleware from the handheld mobile wireless device used at the client node. Considering the capability of the system on which the middleware is maintained, this overhead is negligible on the middleware side too.

## 7.7 Summary

Performance evaluation is essential for ensuring efficiency and effectiveness of any model. The proposed model is evaluated based on five critical parameters (turnaround time, content density, parsing time, transmission time, and processing overhead).

Table 7.4: The proposed transaction model: Objectives Vs Accomplishments

| **Criterion** | **XML/SOAP based** | **LXML/SOA based (Proposed)** |
|---|---|---|
| Lightweight | No | Yes |
| Reliable | Yes | Yes |
| Suited for distributed environments | Yes | Yes |
| Pagination | Not available | Available |
| Support for disconnectedness | No, require additional mechanisms | Yes |
| Verbosity | High | Low (LXML have 40 - 48%, 35 - 55% advantage over XML and JSON, respectively) |
| Verbosity of schema | High | Low (LXML Schema have 80 - 90% advantage over XML Schema) |
| Offloading of middleware from mobile device to server | Yes, but not dynamic | Yes |
| Transmission time | High | Low (LXML has 51 - 61% advantage over XML and 15 - 40% advantage over JSON) |
| Processing and parsing overhead | High | Low |

The performance evaluation of the transaction model reveals its robust capabilities in handling data efficiently. It excels in parsing data swiftly, maintaining low processing overhead, and expediting data transmission (see Table 7.4). The evaluation underlines the suitability of the model in business communication. These performance advantages are mainly because of the use of compact LXML as the data interchange format in the proposed transaction model.

Chapter 8

# Mechanisms to Ensure Secure Transaction of LXML Data

## 8.1 Introduction

Security in data transmission is always a major concern due to many reasons such as:

(a) To protect sensitive and confidential data, ensure its privacy and prevent unauthorized access that can lead to data theft and misuse

(b) To prevent intentional or unintentional data corruption and modifications by unauthorized users during transmission

(c) To avoid financial loss and reputational damage due to data breach during transmission

(d) To maintain business confidentiality

## 8.2 Possible Attacks on XML / SOAP Messages

XML and SOAP messages are the most commonly used standards for data exchange between applications using web services. Web services are widely used for application integration and play an important role in B2B communications by providing a standardized, platform independent way of data exchange. It supports communication and seamless data exchange among applications running on different platforms and using different technologies.

Web services are not immune to a wide variety of threats and attacks during transmission in the network. In this chapter, proposals to check rewriting attacks are discussed, with a view to strengthen the transaction model proposed in Chapter 5.

A tolerant and reliable scheme is essential for maintaining integrity and confidentiality during data exchange. Though, it is practically impossible to safeguard data from intentional attacks, proper data validation, sanitization, parsing, using security standards like encryption and signature can mitigate threats associated with XML and SOAP messages [130][132][173].

## 8.3 Rewriting Attacks

Rewriting attacks are a collection of security vulnerabilities that target applications using XML based data interchange format. They are intentional or deliberate injection of irrelevant and malicious data elements to the XML structure without affecting the signature.

Since the LXML format is derived from the XML, they are also vulnerable to rewriting attacks. This chapter proposes a layered mechanism to isolate rewriting attacks in LXML messages.

## 8.4 The Proposed Security Model

The proposed solution for detecting and isolating rewriting attacks in LXML messaging can be subdivided into three layers: (i) LXML Schema creation, (ii) LXML message generation, and (iii) message encryption.

### 8.4.1 LXML Schema Creation

A LXML Schema is created for each and every transaction carried out between the server and the client devices [148]. A system dealing with multiple transactions will have multiple LXML Schemas and each LXML Schema is a mutual agreement between communicating parties on the structure and organization of the data exchange. So every LXML message is created and transmitted according to the format underlined by its LXML Schema. Any incoming or outgoing transactions not confirming to the LXML Schema creates issues during validation and parsing phase and leads to rejection of the message.

As LXML Schemas are the blueprint for generating LXML messages, the root tag name used in the LXML Schema should be exactly the same as in the LXML document root tag. This is to identify each transaction in a system that handles multiple transactions. There should be one to one correspondence between the tags and blocks in the document being transacted and the corresponding LXML Schema. LXML Schema does not contain any data and repetitions of blocks.

The LXML Schema is mainly used to define the mapping rules related to a LXML document. It provides the details such as root tag name, inner tags names, details of nested tags, count of child nodes, details of child tags and their order, attributes defined for each tag. A sample LXML message and its corresponding LXML Schema are shown in Table 8.1.

The LXML Schema defined for an LXML transaction is independent of the document size, and count or number of records contained in the document. It is not altered in normal cases and needs changes only in situations where the transaction is modified to include or delete tags. As every transaction should strictly adhere to its LXML Schema, unauthorized inclusions and deletions of tags can be easily identified by comparing with the LXML Schema.

Table 8.1: An example of LXML document (a) and its corresponding LXML Schema (b)

**LXML Message**

```
invoice
<00>
<01>ABC Corp
<01>Series A
<01>Delhi
<00 childCount=100>
<01>
   <02>A01
   <02>1400
   <02>15
<01>
   <02>A02
   <02>1450
   <02>10
```

**LXML Schema**

```
<invoice>
    <manufacturer>
        <name></name>
        <brand></brand>
        <address></ address>
    </manufacturer>
    <products childCount=100>
        <product>
              <name></name>
              <price></price>
              <units></units>
        </product>
    </products>
</invoice>
```

## 8.4.2 Message Creation

Every LXML message has two sections namely Message Header (also known as the header) and Message Body (also known as the message). The overall structure of the LXML message is as shown in Figure 8.1.

The header contains vital information required from transmitting the message to the destination such as destination address, source address to identify the sender, and sequence number. In addition to the above fields, three more fields are included to ensure secure transmission between the devices: (i) checksum, (ii) timestamp in the header, designated as TH, and (iii) digital signature

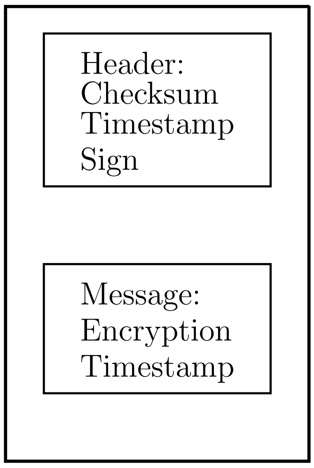

Figure 8.1: Format of LXML message

The checksum holds the CRC checksum computed for the header. The message body is not considered for calculating the header checksum. The EPOCH timestamp of message generation is kept in the timestamp field. A timestamp field is intentionally kept in the message body and holds the same value as in the message header. This value is kept to identify message alterations. Any change or mismatch in the EPOCH timestamp indicates alterations in the header. The timestamp kept in the header is to minimize the probability of false hits during signature creation / validation in the later stage. The last field in the message header is the signature field that holds the 64 bit digital signature.

The message body has included a timestamp field and a childcount attribute to guard the message from attacks during transmission. The message body is created in accordance with the LXML Schema. LXML Schema holds the details of tags, attributes and nested container blocks available in the LXML message. Even if the message has a digital signature, an intruder can make modifications that are undetected [134][145], as mentioned in Section 2.6.

To guard from such attacks, there has to be a mechanism to record the number of children within a container tag. To serve this purpose, a childcount attribute is added to each parent tag where nesting appears. Each container tag holds a childcount attribute that specifies the number of children for that tag and is populated during message creation. This will remove the possibility of "undetected" modifica-

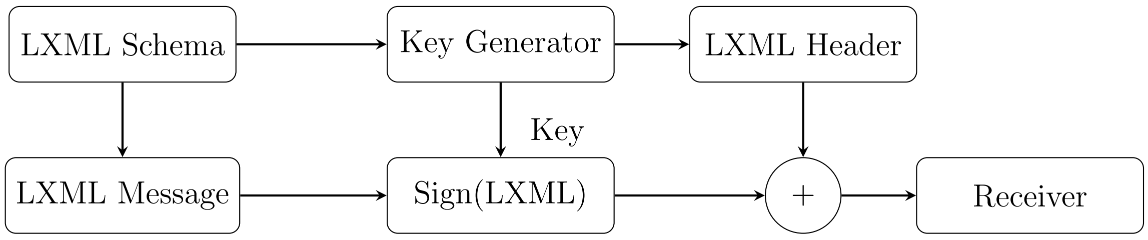

Figure 8.2: Representation of encryption process at the sender

tions, as any mismatch in the actual number of child tags and the childcount will expose the occurance of attacks.

The LXML Schema generated for an LXML transaction defines the mapping rules and has a virtual role in parsing and extracting the document. Before parsing, the LXML document is validated against the LXML Schema. Any inclusions, deletions, alterations of tags or mismatches can easily be detected. In addition to message content, an EPOCH timestamp field is included. This timestamp field auto populates the time of the last content updation and that value is shared with the message header as well.

### 8.4.3 Message Encryption

The main focus of this layer is to prevent unauthorized modifications of contents in the LXML message. This layer utilizes standard encryption and decryption techniques. Each transaction is encrypted at the source end and decrypted at the destination end using AES encryption / decryption algorithm with a shared key. One peculiarity is in the way the key is chosen. The key is computed for each transaction as a function of corresponding LXML Schema. This ensures that the key will be different for different transactions.

Unauthorized alteration of LXML messages can be detected using signature validation after message decryption. The block diagram representing the key generation and signing is shown in Figure 8.2.

To protect the LXML messages from the rewriting attacks, following actions are performed at the sending and receiving ends.

At the sender's end:

- Construct LXML message
- Insert the EPOCH timestamp T to the message
- Generate LXML signature of the message and keep in the header
- Encrypt the entire message using AES encryption algorithm with a shared key K. The key is different for each transaction and is computed based on the LXML Schema of the LXML message

  Encrypted Message = AES (Sign (LXML+T) + K)
- Send the encrypted message to the destination

At the receiver's end:

- Receive the message from the source in encrypted form
- Decrypt the message using the shared key K (the same key used for encryption)

  Decrypted Message = AES (Sig (LXML+T) + K)
- Extract the timestamp and recalculate the LXML signature to ensure that no unauthorized modifications occurred to the message.

  LXML + T = Sign (Decrypted Message)
- Check whether there is a mismatch of T and TH (time stamp in the header). If yes, there is an unauthorized editing of the message; the message gets rejected.

In case the device is a resource constrained device, it can choose any combination of these layers at the sending and receiving ends.

## 8.5 Case Studies

The proposed model enforces security in different layers and each layer contributes to the overall system security. This section examines how the system handles major types of rewriting attacks.

### 8.5.1 Scenario 1: Prevention of Redirection Attacks

In redirection attacks, the attackers intentionally direct the request to some other malicious locations or pages. This is a common attack in financial transactions where the user is directed to some fraud locations with an intention to detain personal information like password. URL redirection attacks constitute about 17% of the total cyber infections [173][174].

This intentional URL redirection attacks can be resisted using the header checksum and timestamp fields associated with the LXML message. A checksum is computed for the header part of the message including the timestamp and excluding the data contents. Same timestamp is kept in the content as well. The whole message is digitally signed and transmitted to the destination. At the destination end, the checksum is recalculated and the time stamp values are extracted. Any change in the checksum or mismatch in the EPOCH timestamp reflects alterations in the header. A sample header message is displayed in Table 8.2.

The timestamp is kept in the header in order to minimize the probability of false hits during signature creation / validation in the later stages. Since the same time stamp is used in both header and message, any attacker changing the header will be forced to tamper the message content to achieve his objective. But any such attempt to modify the content is checked with the help of encryption techniques used in the next level.

### 8.5.2 Scenario 2: Prevention of Adding / Removing tags in LXML

Adding / removing tags in LXML messages is another type of rewriting attack. Usually, such types of attacks occur due to the lack of validation or sanitization of input data. Attackers may inject or remove elements to gain unauthorized access to sensitive data.

The use of LXML Schema can withstand these types of vulnerabilities. The LXML messages are created in accordance with the LXML Schema in which the structure, organization and mapping rules of the LXML message are defined. The

Table 8.2: Sample header of an LXML message for request and response

| **Request** | **Response** |
|---|---|
| `POST /<DEVICE ADDRESS>HTTP/1.1`<br>`Content-Type:application /x-www-`<br>`form-urlencoded`<br>`Connection:close`<br>`Via: ANDR_GTWAY..Accept:*/*`<br>`Host:127.0.0.1:8080`<br>`Content-Length:1024`<br>`Chksum: 0x01FF`<br>`TS:1672322475`<br>`SEQNO:01`<br>`SIGN:mJCK/y5DTBauu7EFqKCPgUylIzY8hSHph`<br>`UIGfphEEl5KfLcXTTEQ4apebrf5Cu5Pujzcwp4`<br>`D4an4pje+ZtHt6hIY3rFSAKoidhWGyW+734A=`<br>`<MESSAGE>` | `HTTP/1.1 200 OK`<br>`Content-Type: text/plain`<br>`Date: Thus, 29 Dec 2022`<br>`19:29:04 GMT+05:30`<br>`Content-Length:256`<br>`<MESSAGE>` |

LXML message maintains exact tag names, attribute names, and their count as given in the corresponding LXML Schema. LXML Schema is an aid for validating, parsing, extracting and processing the data contained within the document. Adherence to the corresponding LXML Schema is strictly maintained while creating the message.

Upon receiving the LXML message at the destination, the message is validated using the same LXML Schema that was used to generate the message. Any violation observed during the validation process indicates a security breach and leads to the rejection of the message.

Table 8.3 illustrates a LXML message for an invoice transaction and its corresponding LXML Schema. From the LXML Schema, it can be inferred that (a) all incoming invoice LXML message has two nested tags at level 0: manufacturer and

Table 8.3: XML message, its LXML equivalent and LXML Schema

| **XML Format** | **LXML Format** | **Data Binding** |
|---|---|---|
| `<invoice>` | | |
| `<manufacturer>` | | |
| `<name>ABC Corp</name>` | | |
| `<brand>series A</brand>` | `invoice` | `<invoice>` |
| `<addr>Delhi</addr>` | `<00>` | `<manufacturer>` |
| `</manufacturer>` | `<01>ABC Corp` | `<name></name>` |
| `<products>` | `<01>Series A` | `<brand></brand>` |
| `<product>` | `<01>Delhi` | `<addr></ addr>` |
| `<name>A01</name>` | `<00 childCount=2>` | `</manufacturer>` |
| `<price>1400</price>` | `<01>` | `<products childCount=2>` |
| `<units>15</units>` | `<02>A01` | `<product>` |
| `</product>` | `<02>1400` | `<name></name>` |
| `<product>` | `<02>15` | `<price></price>` |
| `<name>A02</name>` | `<01>` | `<units></units>` |
| `<price>1450</price>` | `<02>A02` | `</product>` |
| `<units>10</units>` | `<02>1450` | `</products>` |
| `</product>` | `<02>10` | `</invoice>` |
| `</products>` | | |
| `</invoice>` | | |

products, (b) the manufacture tag is a simple container tag with three fields – name, brand and address, (c) the products tag is a repeating block having 3 simple tags – name, price and units, and (d) the message may contain zero or more products as specified in the childcount attribute.

Any incoming invoice transaction not satisfying the above structure is considered as malicious and rejected during validation. To elude the validation process, all

```
<invoice>
   <manufacturer>
      <name></name>
      <brand></brand>
      <addr></ addr>
   </manufacturer>
   <products>
      <product>
         <name></name>
         <price></price>
         <units></units>
      </product>
   </products>
</invoice>
```

```
invoice
<00>
<01>ABC Corp
<01>series A
<01>St No 1
<00>
<01>
  <02>A01
  <02>1400
  <02>15
  <02>X1 <-- injected
<01>
  <02>A02
  <02>1450
  <02>10
  <02>X2  <-- injected
```

Figure 8.3: (a) Untampered LXML Schema (b) LXML message with additional tag injected

attempts to add or delete tags – whether of simple or nested tags, the intruder needs to make corresponding changes in the LXML Schema as well. Since the sender and receiver have a mutual agreement about the LXML Schema, such a change is not possible. Any such attempts can be easily detected and isolated at the destination end. A sample altered LXML message and the untampered XML Schema is depicted in Figure 8.3.

The structural and document organizational validation in LXML documents can be easily detected using the LXML Schema. But addition or removal of a whole block in a repeated container block cannot be traced by comparing the tampered message with the LXML Schema.

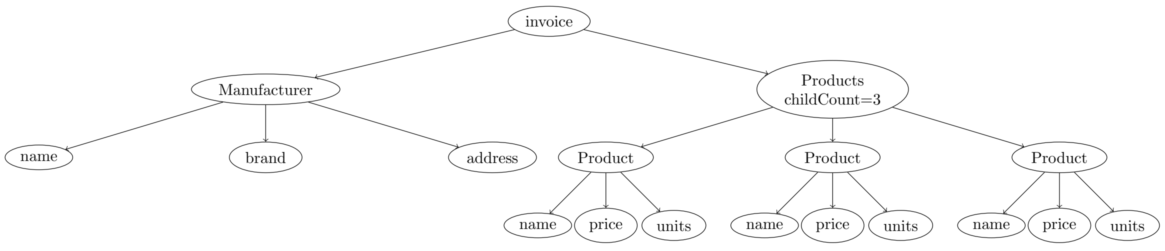

Figure 8.4: LXML Message representation displaying childcount attribute

### 8.5.3 Scenario 3: Prevention of Injecting an Additional Block in a Nested Tags

Intruders can inject their records to the system by populating a block in nested tags. In the invoice transaction given above, a fake product can be inserted to the system by adding a product block to the invoice. Similarly, they can remove particular records by deleting a block in the nested hierarchy. Both these block insertions and deletions will be unnoticed and pass through the validation process easily as it strictly adheres to the LXML Schema.

To counter such attacks, a childcount attribute is placed in each parent tag where nesting appears. Any mismatch in the actual number of child tags and the childcount can expose such fake insertion and deletion of blocks. Messages with count mismatch will be discarded during LXML processing itself and retransmission requests are generated by sending negative acknowledgements (Figure 8.4).

### 8.5.4 Scenario 4: Prevention of Altering / Modifying the Data without Affecting Tags

In this scenario, the attackers modify the data contained in the document without altering the header information and the document structure. These changes do not alter the LXML elements or their organization and it will be unnoticed during the validation process. Such attacks are performed by both in-house and foreign attackers with the intention to include false information to the system.

This content alteration can be carried out in two ways: (i) altering or modifying the message by inserting false, corrupt or unexpected values to certain tags, and (ii) relocating a tag or a group of tags to another location in the same transaction. Changing the contents of LXML elements can damage the data integrity and reliability. Relocating data elements may affect the semantics of the message and adversely affect the parsing and processing of the message.

All these kinds of content alteration can be detected using signature validation after decryption. A mismatch in computed and available content signature implies such attacks. Any content relocation can be identified by comparing the message with LXML Schema before parsing and processing of the transaction.

The actions performed to identify and isolate rewriting attacks in LXML messages can be summarized as follows:

(i) The header of the messages contains a checksum to avoid header rewriting and redirection attacks.

(ii) The LXML Schema represents the structure and organization of the messages and helps to identify inclusion of additional tags, reordering and relocation of LXML messages.

(iii) The childcount attribute is included in the parent tags of all nested repeating blocks. This count can identify inclusion and removal of additional blocks in nesting.

(iv) The whole LXML body is encrypted instead of sending an open LXML message to prevent unauthorized access.

## 8.6 Performance Evaluation

This section discusses the performance evaluation of the security measures to prevent rewriting attacks in LXML messages. A discussion on the additional processing, and storage requirements incurred to utilize the model is done. Lastly, whether the inclusion of additional mechanisms in the LXML Schema and LXML messages will hinder the performance advantage of LXML is also checked.

Implementing the security measures to prevent the rewriting attacks invite two types of additional overheads: (i) additional processing and time required for encryption and decryption, and (ii) storage space requirements. The proposed model is evaluated based on these criteria.

### 8.6.1 Overhead for Encryption and Decryption

The time requirements for secured transmission of LXML messages can be subdivided into three parts: (i) time required to encrypt and sign the LXML message before sending, (ii) the time required for decrypting a LXML message received, and (iii) validating the signature at the receiver. Hence, the total time requirement at the source end will be the sum of the time taken for LXML message creation, signature generation, and encryption. Similarly, the time taken at the destination end will be the sum of the time taken for decryption, signature validation and parsing the message.

A test bed consisting of a laptop (core i3 processor with 4GB RAM, Microsoft Windows 10 Operating System) and an Android mobile phone (Android Version 9) is used to evaluate the proposed model. The laptop has been configured with JDK version 1.8 to support the development. Java Cryptography Architecture (JCA) API, and Advanced Encryption Standard (AES) with a key size of 128 bits is used for encryption.

The time taken for encrypting and decrypting LXML messages with varying number of tags [175][176] is computed. The observed values are graphically presented in Figure 8.5. The time consumed (in milliseconds) for encryption and decryption is plotted on Y axis and the number of tags are shown in X axis. From Figure 9.8, it is observed that the time required to encrypt the LXML message with 10 tags and 20 tags is 102ms, and 114ms, respectively. The time required to decrypt the LXML message with 10 tags and 20 tags is 124ms and 138ms, respectively. The experiment is repeated up to 50 tags. It can be inferred that irrespective of the increase in size in double for tags, the corresponding change in the time taken for encryption and decryption is, in the range of 7 to 11% and 8 to 11%, respectively. The time

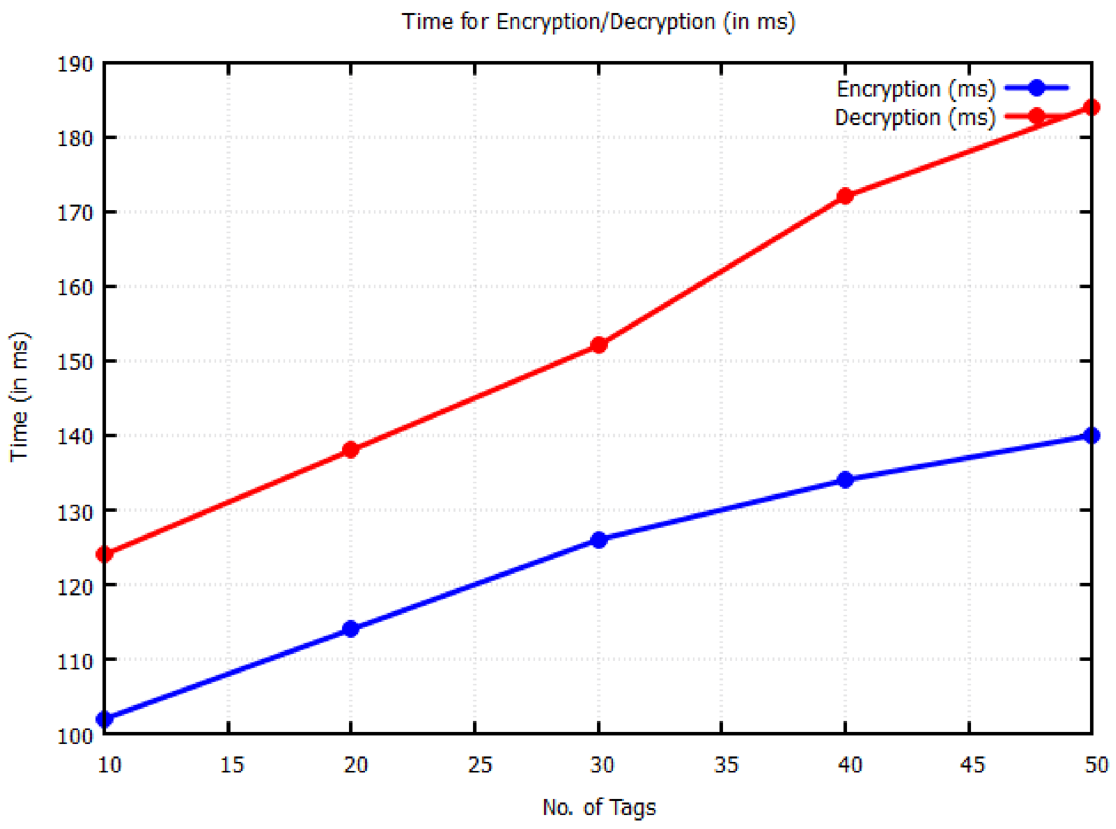

Figure 8.5: Time consumed for encrypting and decrypting LXML (along Y axis) Vs number of tags (along X axis)

requirement for encryption - decryption is dependent on the algorithm chosen and the processing capacity of the system [175].

It can be inferred that the additional overhead incurred due to encryption and decryption of LXML message is, hence negligible. Also the time required for encrypting and decrypting SOAP messages are evaluated for comparison. It is found that the time consumed for encryption and decryption of LXML is comparatively lesser than that of SOAP messages with the same number of tags as illustrated in Figure 8.6 and 8.7. It can be seen that the performance gap between LXML and SOAP messages is widening with the increase in the number of tags. This advantage is clearly due to the compact nature of LXML messages.

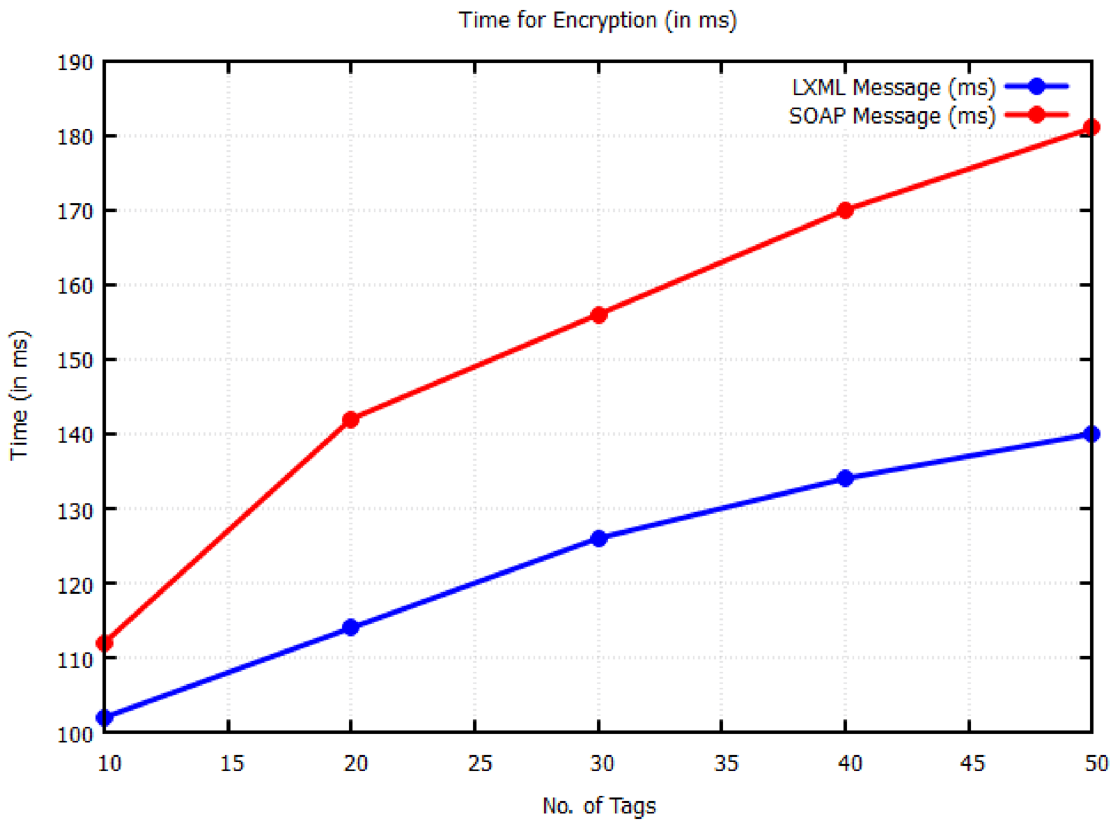

Figure 8.6: Encrypting time of SOAP and LXML messages (along Y axis) Vs number of tags (along X axis)

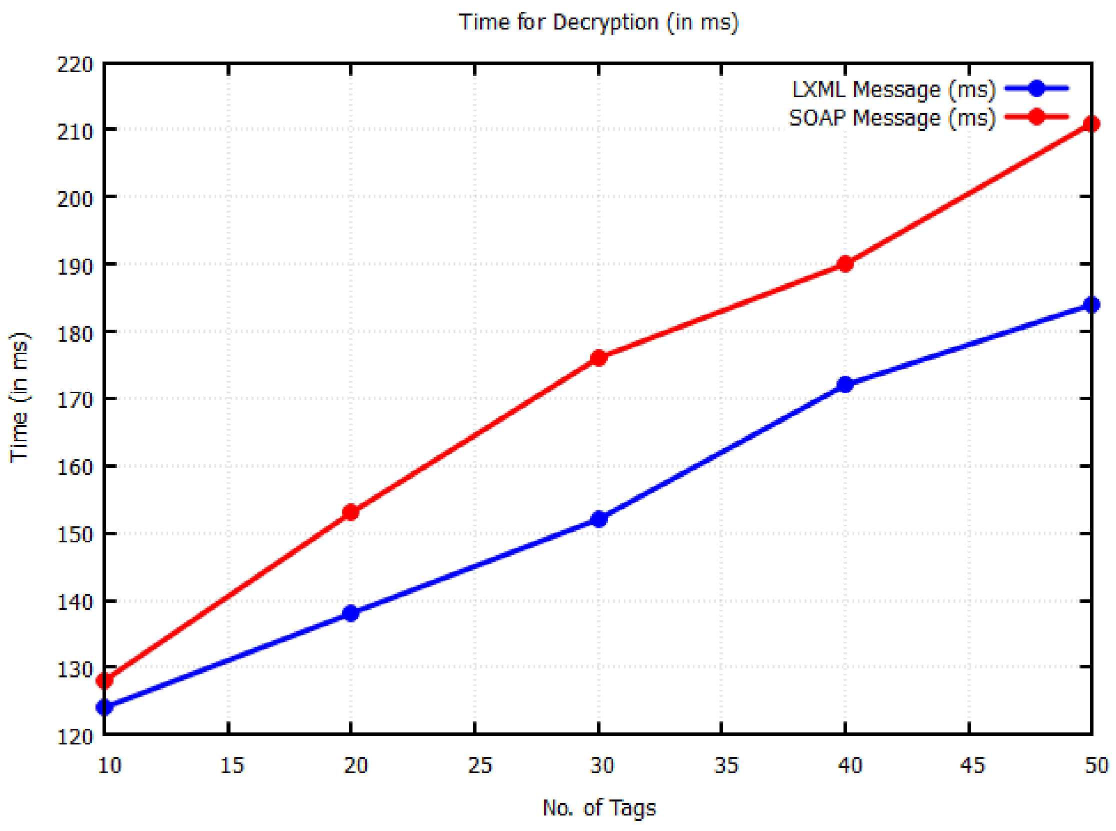

Figure 8.7: Decrypting time of SOAP and LXML messages (along Y axis) Vs number of tags (along X axis)

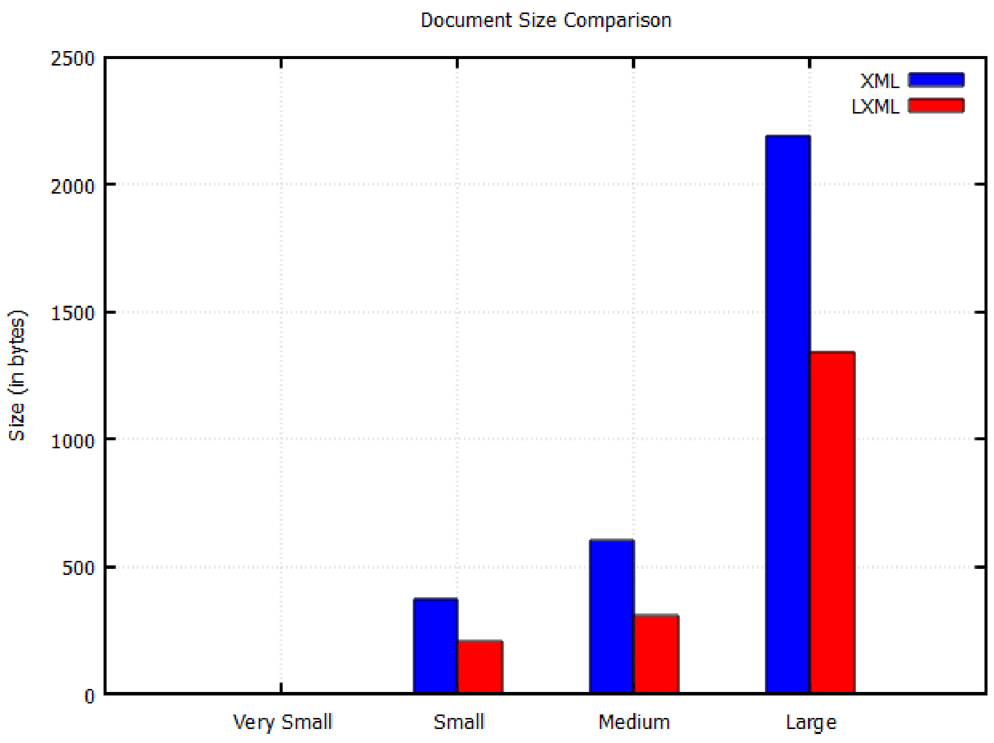

Figure 8.8: Size comparison of XML and LXML for various number of objects

### 8.6.2 Storage Space

LXML Schema and LXML message will contain few additional tags to prevent all kinds of rewriting attacks. It is a concern whether the additional tags and information included for ensuring security in transmission increases the size of the message being transmitted. To analyse the storage requirements, the size of the LXML message is compared with that of the XML (SOAP) message after adding the signature and other security patching.

For this evaluation, sample XML and LXML messages with varying numbers of objects (ranging from 100 to 9000 objects) are considered. The results obtained are plotted in Figure 8.8. The sizes of the XML and LXML schema are also compared as illustrated in Figure 8.9. It can be inferred from Figure 8.8 and 8.9 that LXML has a clear advantage over XML in terms of sizes of message and schema. It can also be inferred that the inclusion of security tags does not sacrifice the performance advantages of LXML as a messaging format.

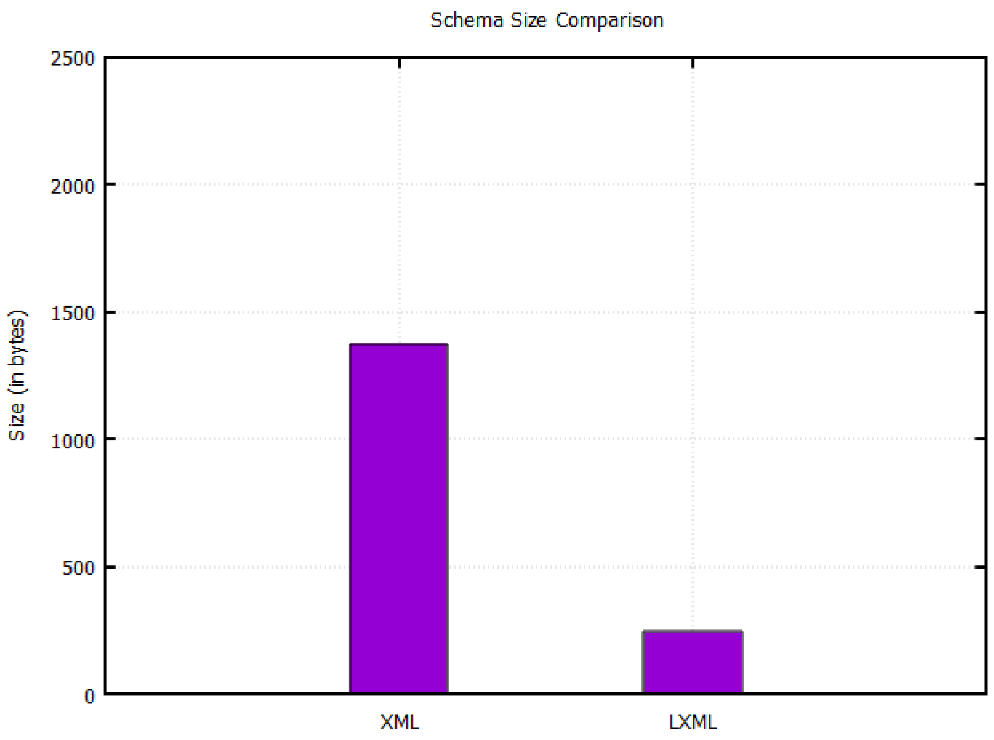

Figure 8.9: Size comparison of XML and LXML Schema

## 8.7 Summary

This chapter outlines the security mechanisms employed to prevent rewriting attacks on LXML messages while in transit. It can be concluded that the layered approach with creation of LXML Schema, creation of message with four additional fields (checksum, timestamp in the header, childcount and digital signature) and encryption of the message can successfully counter rewriting attacks in LXML message transactions. Also, it is experimentally proved that the mechanisms implemented to ensure security measures to check rewriting attacks does not affect the overall efficiency of the LXML messages (see Table 8.4). Hence the proposed work strengthens the foundations for LXML as an alternative data interchange format, and the proposed transaction model based on LXML.

Table 8.4: Measures to check rewriting attacks: Objectives Vs Accomplishments

| **Criterion** | **XML** | **LXML** |
|---|---|---|
| Mechanisms to prevent rewriting attacks | Yes | Yes |
| Security holes reported in literature | Yes | No (all reported holes about rewriting attacks are patched) |
| Detects all kinds of rewriting attacks in the literature | No | Yes |
| Verbosity of structures to prevent rewriting attacks | High | Low |
| Verbosity of schema | High | Low |
| Processing overhead for encryption and decryption | 18 - 29% for encryption and 9 - 18% for decryption | 7 - 11% for encryption and 8 - 11% for decryption |
| Offloading of middleware from mobile device to server | Yes (but not dynamic) | Yes |
| Additional storage space requirements | Verbose | Negligible |

Chapter 9

# Conclusions

The rapid penetration of mobile devices has created numerous opportunities and challenges for users, developers, researchers, and infrastructure designers. Affordable Internet access has spurred the widespread use of wireless networks across various sectors, making mobile devices integral for resource-intensive applications such as ERP and CRM.

Anywhere computing with wireless mobile devices is characterized by resource constraints, relocation, and mobility. A transaction model in mobile computing is necessary to maintain the consistency and integrity of data in the wireless mobile environment. To maintain data consistency and integrity, a reliable, secure, paginated, disconnected, and distributed transaction model using the lightweight LXML format is proposed; the format is experimentally evaluated, proving its relevance in the current scenario.

Although mobile devices are increasingly resource-rich, conserving resources remains crucial. XML and JSON, while popular, are verbose and resource-intensive. This thesis proposes a lightweight data interchange format, LXML, which retains XML's features (extensibility, readability, schema awareness, and platform neutrality) but with less verbosity. Performance analysis shows LXML reduces message size by 40 - 48% compared to XML and 35 - 55% compared to JSON. LXML Schema also achieves an 80 - 90% size reduction compared to XML Schema. Similar advantages are reported in favour of LXML in content density, parsing, serialization/deserialization, marshalling/unmarshalling, and transmission times.

The proposed transaction model, based on LXML, is lightweight, reliable, paginated, and supports transactions in disconnected and distributed wireless mobile environments. A transaction model is proposed by integrating LXML with SOA. The model comprises three layers: client agent, middleware, and host. Reliability is ensured through packet delivery with acknowledgment and sequencing, while disconnectedness is managed by resuming message delivery post-network unavailability. Lightweightedness is achieved using LXML and offloading resource-intensive tasks to a server. Pagination allows large messages to be segmented and reassembled. Performance evaluation shows advantages in upload/download transmission time, LXML-Web service conversion overhead, and content density.

LXML is vulnerable to rewriting attacks, which can cause unauthorized message modifications. To prevent these, a layered architecture is proposed, ensuring secure message transmission with minimal processing and memory overhead. Case studies show that this approach effectively counters various rewriting attacks, making the transaction model suitable for secure transmission of data.

The proposed model poses the following demerits:

(i) Processing overhead due to middleware: The model utilizes a middleware layer to perform data conversions as the LXML format is not compatible with existing servers. There is a processing overhead for such conversion in the middleware, though it is negligible. This overhead can be completely avoided when the server becomes LXML friendly and exposes LXML as part of their services instead of XML or JSON.

## 9.1 Future Research Directions

The proposed transaction model, based on the less verbose LXML format, enhances XML with reduced document size and improved transmission efficiency, making it ideal for the shift of business models to mobile platforms. Integrating LXML with transaction models can significantly improve data transmission and processing efficiency.

The fundamental objective of any data interchange format is to improve transmission efficiency without sacrificing data integrity or interoperability. The proposed format and its usage face many challenges.

### 9.1.1 Standardization and Compatibility

LXML is not yet a standard data interchange format and lacks support from CRMs and other servers. The challenge is to establish LXML as a standard by defining clear specifications, developing supporting libraries and tools, providing implementation guidelines, and addressing versioning and compatibility issues to facilitate smooth data exchange.

### 9.1.2 Interoperability and Integration

Another challenge is to enhance LXML with semantic interoperability; this will enable systems to understand and interpret data meaningfully. This could be achieved by incorporating ontologies, semantic annotations, and knowledge graphs into the format.

## 9.2 Further Enhancements

(a) Improving efficiency: Though LXML is expected to be considered as the most promising approach for data interchange, there exists scope for further improvement.

(b) Streamlined serialization and deserialization: Improving serialization and deserialization efficiency is crucial for data transmission. Developing new algorithms and mechanisms for optimized serialization and deserialization can significantly enhance processing speed, reduce parsing time, and minimize resource requirements during transmission.

(c) Integration with emerging technologies: Researching the integration of LXML format with emerging technologies like IoT, smart home/office systems, and

Artificial Intelligence is another important area. This involves exploring how LXML can support efficient data transmission in decentralized, heterogeneous, and unreliable systems, as well as how it can be seamlessly integrated into AI-driven data processing workflows.

(d) Security and privacy: Research in security continues to advance, particularly in encryption, data anonymization, and access controls, which are critical for safeguarding data. Incorporating these advancements into the LXML format is essential to address emerging threats and potential security vulnerabilities, emphasizing the ongoing need for research in this area.

Chapter 10

# Recommendations

The work entitled "A Lightweight, Reliable, Secure, Paginated, Distributed, and Disconnected Message Transaction Model for Wireless Mobile Environment" is centred on a novel data interchange format known as LXML. The format is a less verbose extension of XML format. The proposed model has significant implications across diverse fields and applications.

1. Efficiency in data transfer: Factors like reduced bandwidth usage and faster data transfer improves the efficiency and is crucial for applications where real-time data updates are essential, such as financial applications, and IoT systems.

2. Resource optimization: The compactness and reduced verbosity bring down the amount of data that needs to be transmitted, resulting in reduced bandwidth usage and storage requirements.

3. Improved performance: Efficiency in serialization and deserialization can lead to faster data parsing and processing leading to improved system performance.

4. Energy efficiency: Faster processing and data transfer conserves the battery power, which is very critical in resource constrained environments.

5. Security and reliability: The model incorporates diverse mechanisms to avoid data breaches and other security threats and to enhance data integrity and confidentiality.

6. Scalability and extensibility: The proposed model is capable of handling bulk volume of data with less resource requirements. The model is extensible and ensures greater flexibility and adaptability.

7. Enhanced user experience: Enhancement in parsing, processing and data transfer reduces user waiting time and latency. Applications adopting this model provide better user experience.

The LXML data interchange format has the potential to enhance the efficiency, cost - effectiveness, and performance of data - driven applications and systems across a wide range of industries. It has the capacity to transform how data is transmitted, processed, and utilized, ultimately contributing to improved user experiences and resource optimization.

Based on the current findings, the model should be implemented in diverse applications areas including business data exchanges, data integrations, data storage, web services and web APIs, big data analytics, IoT and multimedia streaming. In this transaction model, the data is precisely represented, efficiently transmitted, properly interpreted and thereby supports seamless communication between the sender and the receiver. The LXML format has the potential to excel in a diverse field of communication due to its inherent characteristics.

# References

[1] S.K. Madria and B. Bhargava. A transaction model for mobile computing. In *Proceedings. IDEAS'98. International Database Engineering and Applications Symposium (Cat. No.98EX156)*, volume 1, pages 92–102, 1998.

[2] G.H. Forman and J. Zahorjan. The challenges of mobile computing. *Computer*, 27(4):38–47, 1994.

[3] A. Zaslavsky and Z. Tari. Mobile computing: overview and current status. *Journal of Research and Practice in Information Technology*, 30(2):42–52, 1998.

[4] Tomasz Imielinski and B. R. Badrinath. Mobile wireless computing: Challenges in data management. *Commun. ACM*, 37(10):18–28, 1994.

[5] X D Zhou, Arkady Zaslavsky, Rosanne Price, and Ammir Rasheed. Efficient object-oriented query optimisation in mobile computing environment. *Australian Computer Journal*, pages 65 – 75, 1998.

[6] Fred Douglis, Ramon Caceres, M. Kaashoek, P. Krishnan, Kai Li, Brian Marsh, and Joshua Tauber. *Storage Alternatives for Mobile Computers*, volume 353, pages 473–505. Springer, Boston, MA, 2007.

[7] B. Badrinath, Arup Acharya, and Tomasz Imielinski. Impact of mobility on distributed computations. *ACM Operating Systems Review*, 27:15–20, 03 1993.

[8] M. Satyanarayanan, James J. Kistler, Lily B. Mummert, Maria R. Ebling, Puneet Kumar, and Qi Lu. *Experience with Disconnected Operation in a Mobile Computing Environment*, pages 537–570. Springer US, Boston, MA, 1996.

[9] Mohamed Sarwat, Jie Bao, Chi-Yin Chow, Justin Levandoski, Amr Magdy, and Mohamed F. Mokbel. *Context Awareness in Mobile Systems*, pages 257–287. Springer International Publishing, Cham, 2015.

[10] Oystein Sigholt, Besmir Tola, and Yuming Jiang. Keeping connected when the mobile social network goes offline. In *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 59–64, 2019.

[11] Amreen Ayesha and Augustian Isaac. Secure offline/online device to device communication. *International Journal of Control Theory and Applications*, 9(02):1267–1272, 2016.

[12] Gu GQ and Lu J-Z. Some issues of computer networks: architecture and key technologies. *J Comput Sci Technol*, 21(5):708–722, 2008.

[13] Martin Michael. Energy awareness for mobile devices. 01 2005. In: Research seminar on energy awareness. University of Helsinki.

[14] Kangasharju J. Xml messaging for mobile devices. page 255, 2008. Available at http://ethesis.helsinki.fi/ (Accessed on 21 April 2023).

[15] Jaakko Kangasharju, Tancred Lindholm, and Sasu Tarkoma. Xml messaging for mobile devices: From requirements to implementation. *Computer Networks*, 51:4634–4654, 11 2007.

[16] J.A. Paradiso and T. Starner. Energy scavenging for mobile and wireless electronics. *IEEE Pervasive Computing*, 4(1):18–27, 2005.

[17] Tomasz Imielinski and Henry F. Korth. *Introduction to Mobile Computing*, volume 353, pages 1–43. Springer US, Boston, MA, 1996.

[18] Brahim Ghribi and Luigi Logrippo. Understanding gprs: the gsm packet radio service. *Computer Networks*, 34(5):763–779, 2000.

[19] M. C. Bale. Voice and internet multimedia in umts networks. *BT Technology Journal*, 19(1):48–67, 2001.

[20] Ramraj Dangi, Praveen Lalwani, Gaurav Choudhary, Ilsun You, and Giovanni Pau. Study and investigation on 5g technology: A systematic review. *Sensors*, 22(1), 2022.

[21] Anusha Rahul, Gokul Krishnan, Unni H, and Sethuraman Rao. Near field communication (nfc) technology: A survey. *International Journal on Cybernetics & Informatics*, 4(1):133–144, 2015.

[22] IEEE. Ieee standard for information technology–telecommunications and information exchange between systems - local and metropolitan area networks–specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications. *IEEE Std 802.11-2020 (Revision of IEEE Std 802.11-2016)*, pages 1–4379, 2021.

[23] Mohaiminul Islam and Shangzhu Jin. An overview research on wireless communication network. *Advances in Wireless Communications and Networks*, 5(1):19–28, 2019.

[24] C. E. Perkins. *Ad Hoc Networking*, pages 1–385. Addison-Wesley, Boston, Massachusetts, USA, 2001.

[25] David Remondo. *Wireless Ad Hoc Networks: An Overview*, pages 746–766. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[26] Daniel Benevides da Costa and Hong-Chuan Yang. Grand challenges in wireless communications. *Frontiers in Communications and Networks*, 1, 2020.

[27] Pengcheng Zhu, Jiamin Li, Dongming Wang, and You Xiaohu. Machine-learning-based opportunistic spectrum access in cognitive radio networks. *IEEE Wireless Communications*, 27:38–44, 02 2020.

[28] Yu Jin, Jiayi Zhang, and bo ai. Channel estimation for cell-free mmwave massive mimo through deep learning. *IEEE Transactions on Vehicular Technology*, 68:10325–10329, 08 2019.

[29] Kai Yang, Nan Yang, Neng Ye, Min Jia, Zhen Gao, and Rongfei Fan. Non-orthogonal multiple access: Achieving sustainable future radio access. *IEEE Communications Magazine*, 57(2):116–121, 2019.

[30] Boubakr Nour, Kashif Sharif, Fan Li, and Yu Wang. Security and privacy challenges in information-centric wireless internet of things networks. *IEEE Security & Privacy*, 18(2):35–45, 2020.

[31] B. Aiken, J. Strassner, B. Carpenter, I. Foster, C. Lynch, J. Mambretti, R. Moore, and B. Teitelbaum. Rfc2768: Network policy and services: A report of a workshop on middleware. USA, 2000. RFC Editor.

[32] Amirhossein Farahzadi, Pooyan Shams, Javad Rezazadeh, and Reza Farahbakhsh. Middleware technologies for cloud of things: a survey. *Digital Communications and Networks*, 4(3):176–188, 2018.

[33] J. Al-Jaroodi, N. Mohamed, and Hong Jiang. Distributed systems middleware architecture from a software engineering perspective. In *Proceedings of the Fifth IEEE Workshop on Mobile Computing Systems and Applications*, pages 572–579, 2003.

[34] Upkar Varshney and Ron Vetter. Emerging mobile and wireless networks. *Commun. ACM*, 43(6):73–81, 2000.

[35] A. Holzinger, A. Nischelwitzer, and M. Meisenberger. Mobile phones as a challenge for m-learning: examples for mobile interactive learning objects (milos). In *Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications Workshops*, volume 1, pages 307–311, 2005.

[36] Maniar Nipan, Bennett Emily, Steve Hand, and Allan George. The effect of mobile phone screen size on video based learning. *Journal of Software*, 3:51–61, 04 2008.

[37] Sirapat Boonkrong and Dinh Pham Cao. The comparison of impacts to android phone battery between polling data and pushing data. In *Proceedings of the International Conference on Computer Networks and Information Technology (ICCNIT 2013)*, pages 84–89, 06 2013.

[38] Hongyao Luo, Zhichuan Huang, and Ting Zhu. A survey on spectrum utilization in wireless sensor networks. *Journal of Sensors*, 2015:1–13, 2015.

[39] hembelihle Dlamini and Sifiso Vilakati. Remote and rural connectivity: Infrastructure and resource sharing principles. *Wireless Communications and Mobile Computing*, 2021:1–12, 2021.

[40] Mohamed Shameer M C and Abdul Haleem P P. A study on the requirements of a transaction model in mobile environment. *International Journal of Computer Science and Information Technologies*, 13(05):103–109, 2022.

[41] E. Pitoura and B. Bhargava. Dealing with mobility: Issues and research challenges. pages 1–18, 1993. Technical report CSD-TR-93-070.

[42] Amer O. Abu Salem and Ahmad H. Al-Qeerm. Classification of transaction models in mobile database system. In *2015 2nd World Symposium on Web Applications and Networking (WSWAN)*, pages 1–6, 2015.

[43] Ayse Yasemin Seydim. An overview of transaction models in mobile environments. page 12, 2001. Available at https://api.semanticscholar.org/CorpusID:13861597 (Accessed on 23 April 2021).

[44] Abdelsalam Helal, Santosh Balakrishnan, Margaret Dunham, and Ramez Elmasri. A survey of mobile transaction models. 1996. Technical report Paper 1259. Available at https://docs.lib.purdue.edu/cstech/1259 (Accessed on 23 April 2023).

[45] Rahul B. Mannade and Amol B. Bhande. Challenges of mobile computing: An overview. *International Journal of Advanced Research in Computer and Communication Engineering*, 2(1):13109–3114, 2013.

[46] Aleksandar S. What percentage of internet traffic is mobile in 2021?, 2021. Available at https://techjury.net/blog/what-percentage-of-internet-traffic-is-mobile. (Accessed on 10 Nov 2021).

[47] Ericsson Inc. Mobile data traffic outlook, 2023. Available at https://www.ericsson.com/en/reports-and-papers/mobility-report/dataforecasts/mobile-traffic-forecast. (Accessed on 20 Dec 2023).

[48] Newstalk. Mobile devices have overtaken the human population, 2018. Available at https://www.newstalk.com/news/mobile-devices-have-overtaken-the-human-population-499358. (Accessed on March, 2021).

[49] DATAREPORTAL. Digital around the world, 2021. Available at https://datareportal.com/global-digital-overview. (Accessed on 03 March 2021).

[50] Deyan Georgiev. Smartphone statistics, 2021. Available at https://review42.com/resources/smartphone-statistics/. (Accessed on 15 June 2021).

[51] Ambrin Javed, Muhammad Alyas Shahid, Muhammad Sharif, and Mussarat Yasmin. Energy consumption in mobile phones. *International Journal of Computer Network and Information Security(IJCNIS)*, 9(12):18–28, 2017.

[52] Xin Zheng, Yu Nan, Fangsu Wang, Ruiqing Song, Gang Zheng, Gaocai Wang, Yuting Lu, and Qifei Zhao. A data transmission strategy with energy minimization based on optimal stopping theory in mobile cloud computing. *Wireless Communications and Mobile Computing*, 2019:1–11, 2019.

[53] G. P. Perrucci, F. H. P. Fitzek, and J. Widmer. Survey on energy consumption entities on the smartphone platform. In *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*, pages 1–6, 2011.

[54] Jameel Ali and Majid Altamimi. Energy consumption model for data transfer in smartphone. *Computer Communications*, 182:13–21, 2022.

[55] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. Energy consumption in mobile phones: A measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, page 280–293, New York, NY, USA, 2009. Association for Computing Machinery.

[56] Greg Charest. Data exchange mechanisms and considerations, 2022. (Accessed on 31 May 2023).

[57] W3C Recommendation. Extensible markup language (xml) 1.0 (fifth edition), 2008. Available at https://www.w3.org/TR/xml/. (Accessed on 09 May 2023).

[58] Yinglin Li, Lianhe Yang, and Yingzhong Li. Research on apparel cad pattern data interchange format based on xml. In *2009 WRI World Congress on Software Engineering*, volume 3, pages 90–94, 2009.

[59] J. Clark. Xsl transformations (xslt) version 1.0, 1999. World Wide Web Consortium Available at http://www.w3.org/TR/xslt/. (Accessed on 01 May 2023).

[60] J. Clark and S. DeRose. Xml path language (xpath) version 1.0., 1999. World Wide Web Consortium (W3C). Available athttp://www.w3.org/TR/xpath/. (Accessed on 01 May 2023).

[61] D. C. Fallside and P. Walmsley. Xml schema part 0: Primer second edition, 2004. World Wide Web Consortium (W3C). Available at http://www.w3.org/TR/xmlschema-0/. (Accessed on 01 May 2023).

[62] Elliotte Rusty Harold and W. Scott Means. *XML in a nutshell.* O'ReillyVerlag, 2011.

[63] Andrew Fairbanks, James Gribbons, Erik Nybo, David Pean, and Joseph Wright. Research in xml (extensible markup language). *J. Comput. Sci. Coll.*, 17(6):253–254, 2002.

[64] Dan Suciu. *Semistructured Data and XML*, pages 9–30. Springer US, Boston, MA, 2000.

[65] Saurabh Zunke and Veronica D'Souza. Json vs xml: A comparative performance analysis of data exchange formats. *International Journal of Computer Science and Network*, 3(4):257–261, 2014.

[66] P. P. Abdul Haleem and M. P. Sebastian. An energy-conserving approach for data formatting and trusted document exchange in resource-constrained networks. *Knowl. Inf. Syst*, 32(3):559–587, 2012.

[67] Maja Pusnik Bostjan Sumak, Marjan Hericko. Evaluation of xml schema support in knowledge management. *Frontiers in Artificial Intelligence and Applications*, 333:150–160, 2020.

[68] Sang-Kyun Kim, Myungcheol Lee, and Kyu-Chul Lee. Validation of xml document updates based on xml schema in xml databases. In *Database and Expert Systems Applications*, pages 98–108, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[69] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Concepts, Architectures and Applications*, pages 1–354. Springer Berlin, Berlin, Heidelberg, 2004.

[70] Kangasharju J. Mobile xml messaging. page 255, 2008. Technical Report, Available at www.cs.helsinki.fi/u/jkangash/xml-mess.pdf. (Accessed on 16 June 2022).

[71] Nidhi Arora, Savita Kolhe, and Sanjay Tanwani. A comprehensive model to enhance performance of ws-security processing. *International Journal of Computer Science and Mobile Computing*, 5(1):265–270, 2016.

[72] Sarra Abidi, Mehrez Essafi, Chirine Ghedira Guegan, Myriam Fakhri, Hamad Witti, and Henda Hjjami Ben Ghezalai. A web service security governance approach based on dedicated micro-services. *Procedia Computer Science*, 159(1):372–386, 2019.

[73] DaYong Wu, Kien Tsong Chau, JingYi Wang, and ChuTing Pan. A comparative study on performance of xml parser apis (dom and sax) in parsing efficiency. In *Proceedings of the 3rd International Conference on Cryptography, Security and Privacy*, pages 88–92, New York, NY, USA, 2019. Association for Computing Machinery.

[74] Chengkai Li. *XML Parsing, SAX/DOM*, pages 3598–3601. Springer US, Boston, MA, 2009.

[75] Matthias Nicola and Jasmi John. Xml parsing: A threat to database performance. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management, CIKM '03*, page 175–178, New York, NY, USA, 2003. Association for Computing Machinery.

[76] N. Singh and K. Tyagi. A literature review of the reliability of composite web service in service oriented architecture. *JACM SIGSOFT Software Engineering Notes*, 40(1):1–8, 2015.

[77] Yanlei Diao and Michael Franklin. Query processing for high-volume xml message brokering. In *Proceedings 2003 VLDB Conference*, pages 261–272, San Francisco, 2003. Morgan Kaufmann.

[78] Alain Tamayo, Carlos Granell, and Joaquín Huerta. Dealing with large schema sets in mobile sos-based applications. In *Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications, COM.Geo '11*, New York, NY, USA, 2011. Association for Computing Machinery.

[79] JSON. Introducing json, 1999. Available at http://www.json.org/. (Accessed on 01 May 2023).

[80] Bader Alghamdi, Leigh Ellen Potter, and Steve Drew. Identifying best practices in organisational soa governance adoption: Case study of saudi arabia's e-government programme. In *Proceedings of the 20th Pacific Asia Conference on Information Systems, PACIS 2016*, pages 1–14, Chiayi, Taiwan, 2016.

[81] Muzafer H. Saracevic, Emrus Azizovic, and Munir Sabanovic. Comparative analysis of amf, json and xml technologies for data transfer between the server and the client. *Periodicals of Engineering and Natural Sciences (PEN)*, 4(2):76–83, 2016.

[82] Hsun-Ming Lee and Mayur R. Mehta. Defense against rest-based web service attacks for enterprise systems. *Communications of the IIMA*, 13(1):57–68, 2013.

[83] Bakken E. Jorstad I. and Anders Johansen T. Performance evaluation of json and xml for data exchange in mobile services. In *Proceedings of the International Conference on Wireless Information Networks and Systems*, pages 237–240. SciTePres, 2008.

[84] Teng Lv, Ping Yan, and Weimin He. On massive json data model and schema. *Journal of Physics: Conference Series*, 1302(2):22–31, 2019.

[85] Ben Kiki O. Yaml ain't markup language (yaml) version 1.2., 1999. Available at https://yaml.org/spec/1.2/spec.html (Accessed on 01 March 2021).

[86] P P Abdul Haleem and M P Sebastian. An alternative approach for xml messaging. *International Journal of Advanced Research*, 2(1):251–294, 2014.

[87] Yang Yahui. Impact data-exchange based on xml. In *2012 7th International Conference on Computer Science & Education (ICCSE)*, pages 1147–1149, 2012.

[88] P. Greenfield, M. Droettboom, and E. Bray. Asdf: A new data format for astronomy. *Astronomy and Computing*, 12(1):240–251, 2015.

[89] Ahmed K. Elmagarmid. *Database Transaction Models for Advanced Applications*, page 610. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1992.

[90] G.D. Walborn and P.K. Chrysanthis. Supporting semantics-based transaction processing in mobile database applications. In *Proceedings. 14th Symposium on Reliable Distributed Systems*, pages 31–40, 1995.

[91] Calton Pu, Gail E. Kaiser, and Norman C. Hutchinson. Split-transactions for open-ended activities. In *Proceedings of the 14th International Conference on Very Large Data Bases*, VLDB '88, page 26–37, San Francisco, CA, USA, 1988. Morgan Kaufmann Publishers Inc.

[92] Prasanta Kumar Panda, Sujata Swain, and P. K. Pattnaik. Review of some transaction models used in mobile databases. *International Journal of Computer and Communication Technology*, 4(1):311–321, 2013.

[93] Krithi Ramamritham and Panos K. Chrysanthis. *Executive Briefing: Advances in Concurrency Control and Transaction Processing*, page 113. IEEE Computer Society Press, Washington, DC, USA, 1996.

[94] Malasowe Bridget and S. A Idowu. Current state of the art in mobile transaction models: A survey. *International Journal of Advanced Studies in Computer Science and Engineering*, 3(1):22–27, 2014.

[95] Ouri Wolfson. *Mobile Database*, pages 1751–1751. Springer US, Boston, MA, 2009.

[96] Gary D. Walborn and Panos K. Chrysanthis. Transaction processing in pro-motion. In *Proceedings of the 1999 ACM Symposium on Applied Computing*, SAC '99, page 389–398, New York, NY, USA, 1999. Association for Computing Machinery.

[97] Jari Veijalainen, Vagan Terziyan, and Henry Tirri. Transaction management for m-commerce at a mobile terminal. *Electronic Commerce Research and Applications*, 5(3):229–245, 2006. Mobile technology and services.

[98] Bo Chen, David D. Linz, and Harry H. Cheng. Xml-based agent communication, migration and computation in mobile agent systems. *Journal of Systems and Software*, 81(8):1364–1376, 2008.

[99] Kyong-I Ku and Yoo-Sung Kim. Moflex transaction model for mobile heterogeneous multidatabase systems. In *Proceedings Tenth International Workshop on Research Issues in Data Engineering. RIDE 2000*, pages 39–45, 2000.

[100] Yousef J. Al-Houmaily and Panos K. Chrysanthis. 1-2pc: The one-two phase atomic commit protocol. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, SAC '04, page 684–691, New York, NY, USA, 2004. Association for Computing Machinery.

[101] Holliday J., Agrawal D. A., and Abbadi E. Disconnection modes for mobile databases. *Wireless Networks*, 8(1):391–402, 2002.

[102] D. Grigoras. Challenges to the design of mobile middleware systems. In *International Symposium on Parallel Computing in Electrical Engineering (PARELEC'06)*, pages 14–19, 2006.

[103] Agustinus Borgy Waluyo, David Taniar, Wenny Rahayu, and Bala Srinivasan. Mobile service oriented architectures for nn-queries. *Journal of Network and Computer Applications*, 32(2):434 –447, 2009.

[104] A. Schill. Service-oriented architectures: potential and challenges. In *Proceedings of the 15th International Crimean Conference Microwave & Telecommunication Technology*, volume 1, pages 16–18, Sevastopol, Ukraine, 2005.

[105] Gurpreet Kaur and Mohammad Muztaba Fuad. An evaluation of protocol buffer. In *Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon)*, pages 459–462, 2010.

[106] Zhou S, Jadoon W, and Khan IA. Computing offloading strategy in mobile edge computing environment: A comparison between adopted frameworks, challenges, and future directions. *Electronics*, 12(11):24–52, 2023.

[107] Chuan Feng, Pengchao Han, Xu Zhang, Bowen Yang, Yejun Liu, and Lei Guo. Computation offloading in mobile edge computing networks: A survey. *Journal of Network and Computer Applications*, 202(1):103–366, 2022.

[108] Minhaj Ahmad Khan. A survey of computation offloading strategies for performance improvement of applications running on mobile devices. *Journal of Network and Computer Applications*, 56(1):28–40, 2015.

[109] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-oriented computing: State of the art and research challenges. *Computer*, 40(11):38 – 45, 2007.

[110] M. Alodib and B. Bordbar. A model driven architecture approach to fault tolerance in service oriented architectures, a performance study. In *Proceedings of the 12th Enterprise Distributed Object Computing Conference Workshops*, volume 1, pages 293–300, Munich, Germany, 2008.

[111] S. Dagtas, Y. Natchetoi, H. Wu, and L. Hamdi. An integrated lightweight software architecture for mobile business applications. In *Proceedings of the Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA 2008)*, pages 41–50, Vancouver, BC, Canada, 2008.

[112] Renne Tergujeff, Jyrki Haajanen, Juha Leppanen, and Santtu Toivonen. Mobile soa: Service orientation on lightweight mobile devices. In *IEEE International Conference on Web Services (ICWS 2007)*, pages 1224–1225, 2007.

[113] Agustinus Borgy Waluyo, David Taniar, Wenny Rahayu, and Bala Srinivasan. Trustworthy data delivery in mobile p2p network. *Journal of Computer and System Sciences*, 86(1):33 – 48, 2017.

[114] A. Aikebaier and M. Takizawa. A protocol for reliably, flexibly, and efficiently making agreement among peers. *Int. J. Web Grid Serv.*, 5(4):356–371, 2009.

[115] P. Sharma, Z. Xu, S. Banerjee, and S. Lee. Estimating network proximity and latency. *ACM SIGCOMM Comput. Commun. Rev.*, 36(3):35–50, 2006.

[116] A. Qayyum, L. Viennot, and A. Laouiti. Multipoint relaying for flooding broadcast messages in mobile wireless networks. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences*, pages 3866–3875, 2002.

[117] Nurzhan Nurseitov, Michael Paulson, Randall Reynolds, and Clemente Izurieta. Comparison of json and xml data interchange formats: A case study. In *Proceedings of the 22nd International Conference on Computer Applications in Industry and Engineering, CAINE 2009*, pages 157–162, California, USA, 2009. Montana State University.

[118] Peggy Gravitz, Jack Sheehan, and Thom Mclean. Common activities in data interchange format (dif) development. pages 1–2. Report. McLeod Institute of Simulation Sciences, 03 1999. Available at http://www.ecst.csuchico.edu/hla/LectureNotes/99S 177SIWPaper.pdf (Accessed on 30 April 2023).

[119] Jan Mendling. A survey on design criteria for interchange formats. Vienna, Austria, 2004. Department of Information Systems, Vienna University of Economics and Busi-ness Administration. Technical Report JM-2004-06-02. Available at http://wi.wu-wien.ac.at/mendling/publications/TR04-Interchange.pdf (Accessed on 30 April 2023).

[120] James J. Kistler and M. Satyanarayanan. Disconnected operation in the coda file system. *ACM Trans. Comput. Syst.*, 10(1):3–25, 1992.

[121] J.H. Abawajy and M. Mat deris. Supporting disconnected operations in mobile computing. In *IEEE International Conference on Computer Systems and Applications, 2006.*, pages 911–918, 2006.

[122] Sara Bouchenak and Noël de Palma. *Message Queuing Systems*, pages 1716–1717. Springer US, Boston, MA, 2009.

[123] IBM MQ. Introduction to message queuing, 2023. Available at https://www.ibm.com/docs/en/ibm-mq/9.3?topic=overview-introduction-message-queuing (Accessed on 07 Nov 2023).

[124] Guo Fu, Yanfeng Zhang, and Ge Yu. A fair comparison of message queuing systems. *IEEE Access*, PP:1–1, 12 2020.

[125] Amazon Web Services. Benefits of message queues, 2023. Available at https://aws.amazon.com/message-queue/benefits/. (Accessed on 30 September 2023).

[126] D.M. Huizinga and K.A. Heflinger. Experience with connected and disconnected operation of portable notebook computers in distributed systems. In *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*, pages 119 –123, 1994.

[127] D.M. Huizinga and K.A. Heflinger. Two-level client caching and disconnected operation of notebook computers in distributed systems. In *Proceedings of the 1995 ACM Symposium on Applied Computing*, volume 21, pages 390 – 395. Association for Computing Machinery, 1995.

[128] ShivajiKant. Exploring distributed systems: Pull vs. push strategies, 2023. Available at https://medium.com/@shivajiofficial5088/exploring-distributed-systems-pull-vs-push-strategies-c7a31cd47f79 (Accessed on 07 Nov 2023).

[129] Yang Zhao. *A Model of Computation with Push and Pull Processing.* Number UCB/ERL M03/51. Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, 12 2003. Research Project Report, Available at http://www2.eecs.berkeley.edu/Pubs/TechRpts/2003/4192.html (Accessed on 07 Nov 2023).

[130] Navya Sidharth and Jigang Liu. A framework for enhancing web services security. In *31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, volume 1, pages 23–30, 2007.

[131] Irfan siddavatam and Jayant Gadge. Comprehensive test mechanism to detect attack on web services. In *In Proceedings of the 16th IEEE International Conference on Networks*, pages 1–6, 2008.

[132] R.L. Costello. Xml risks and mitigations, 2013. Available at https://www.mitre.org/sites/default/files/pdf/13 2445.pdf. (Accessed on 25 May 2023 ).

[133] A. Nasridinov and Y.H. Byun, J.Y.and Park. A study on detection techniques of xml rewriting attacks in web services. *International Journal of Control and Automation*, 7(1):391–400, 2014.

[134] Michael McIntosh and Paula Austel. Xml signature element wrapping attacks and countermeasures. In *Proceedings of the 2005 Workshop on Secure Web Services, SWS '05*, page 20–27, New York, NY, USA, 2005. Association for Computing Machinery.

[135] Karthikeyan Bhargavan, Cédric Fournet, Andrew D. Gordon, and Greg O'Shea. An advisor for web services security policies. In *Proceedings of the 2005 Workshop on Secure Web Services, SWS '05*, page 1–9, New York, NY, USA, 2005. Association for Computing Machinery.

[136] P.P. Abdul Haleem and M.P. Sebastian. A layered architecture for checking rewriting attacks in resource constrained networks. In *2010 International Conference on Data Storage and Data Engineering*, pages 270–274, 2010.

[137] P.P. Hung, A. Nasridinov, L. Qing, and J.Y. Byun. A solution for injection and rewriting attacks on soap messages in web services security. *Journal of KIISE: Computing Practices and Letters*, 18(3):244–248, 2012.

[138] Karthikeyan Bhargavan, Cédric Fournet, and Andrew D. Gordon. Verifying policy-based security for web services. In *Proceedings of the 11th ACM Conference on Computer and Communications Security, CCS '04*, page 268–277, New York, NY, USA, 2004. Association for Computing Machinery.

[139] A. Benameur, F.A. Kadir, and S. Fenet. Xml rewriting attacks: Existing solutions and their limitations. In *Proceedings of the IADIS International Conference on Applied Computing*, pages 94–102, Algarve, Portugal, 2008. IADIS Press.

[140] Mohammad Ashiqur Rahaman, Maarten Rits, and Andreas Schaad. An inline approach for secure soap requests and early validation. In *Paper presented at European Conference on Open Web Application Security Project (OWASP '06)*, pages 1–115, Leuven, Belgium, 2006.

[141] Sebastian Gajek, Lijun Liao, and Jörg Schwenk. Breaking and fixing the inline approach. In *Proceedings of the 2007 ACM Workshop on Secure Web Services, SWS '07*, page 37–43, New York, NY, USA, 2007. Association for Computing Machinery.

[142] F.A. Kadir. *RewritingHealer: An Approach for Securing Web Service Communication*, page 89. Master Thesis, KTH Royal Institute of Technology, Sweden, 2008.

[143] A. Nasridinov, P.P. Hung, L. Qing, and J.Y. Byun. Xat-soap: Xml-based attacks tolerant soap messages. *Journal of KIISE (Korea Institute of Information Scientists and Engineering): Computing Practices and Letters*, 18(6):489–493, 2012.

[144] Smriti Kumar Sinha and Azzedine Benameur. A formal solution to rewriting attacks on soap messages. In *Proceedings of the 2008 ACM Workshop on Secure Web Services, SWS '08*, page 53–60, New York, NY, USA, 2008. Association for Computing Machinery.

[145] A. Nasridinov, Y.-S. Jeong, J.-Y. Byun, and Y.H. Park. A histogram-based method for efficient detection of rewriting attacks in simple object access protocol messages. *Sec. and Commun. Netw.*, 9(6):492–499, 2016.

[146] Cristian I. Pinzon, Javier Bajo, Juan F. De Paz, and Juan M. Corchado. S-mas: An adaptive hierarchical distributed multi-agent architecture for blocking malicious soap messages within web services environments. *Expert Systems with Applications*, 38(5):5486–5499, 2011.

[147] C.L. Fang, D. Liang, F. Lin, and C.C. Lin. Fault-tolerant web services. *Journal of System Architecture*, 53(1):23–38, 2007.

[148] M.C. Mohammed Shameer, P.P. Abdul Haleem, and Y.K. Puthenpediyakkal. A lightweight data exchange format for mobile transactions. *International Journal of Computer Network and Information Security(IJCNIS)*, 15(3):47–64, 2023.

[149] A. Ekelhart, S. Fenz, M. Goluch, G.and Steinkellner, and E. Weippl. Xml security - a comparative literature review. *Journal of Systems and Software*, 81(10):1715–1724, 2008.

[150] Vincent C. Emeakaroha, Philip Healy, Kaniz Fatema, and John P. Morrison. Analysis of data interchange formats for interoperable and efficient data communication in clouds. In *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, pages 393–398, 2013.

[151] Hwang Ho Kim, Moon Kyung Kim, Jin Young Choi, and Gi-Nam Wang. Design and application of data interchange formats (difs) for improving interoperability in sba. In *Proceedings of the Winter Simulation Conference, WSC '12*, Berlin, Germany, 2012. Winter Simulation Conference.

[152] Kazuaki Maeda. Performance evaluation of object serialization libraries in xml, json and binary formats. In *2012 Second International Conference on Digital Information and Communication Technology and it's Applications (DICTAP)*, pages 177–182, Bangkok, Thailand, 2012. IEEE.

[153] White G, Kangasharju J, Brutzman D, and Williams S. Efficient xml interchange measurements note, 2007. Available at http://www.w3.org/TR/exi-measurements. (Accessed on 20 May 2020).

[154] Jan Stenberg. Xml can give the same performance as json. Available at https://www.infoq.com/news/2013/08/xml-json-performance/. (Accessed on 16 March 2021).

[155] Jackson. Jackson json processor. Available at http://jackson.codehaus.org. (Accessed on 9 May 2020).

[156] Suchit A. Sapate. Effective xml compressor: Xmill with lzma data compression. *International Journal of Education and Management Engineering (IJEME)*, 9(4):1–10, 2019.

[157] IBM. Mobile computing. Available at https://www.ibm.com/topics/mobile-technology. (Accessed on 9 May 2020).

[158] Li Jiang. Application research of xml parsing technology based on android. *Journal of Physics: Conference Series*, 1314:12–15, 10 2019.

[159] JSON. Json serialization usage. Available at http://flexjson.sourceforge.net/. (Accessed on 28 May 2020).

[160] Guanhua Wang. Improving data transmission in web applications via the translation between xml and json. In *2011 Third International Conference on Communications and Mobile Computing*, pages 182–185, Qingdao, China, 2011. IEEE.

[161] XStream. Xstream. Available at http://xstream.codehaus.org/. (Accessed on 17 May 2020).

[162] Google. google gson - a java library to convert json to java objects and vice-versa. Available at http://code.google.com/p/google-gson/. (Accessed on 15 May 2020).

[163] Tommi Aihkisalo and Tuomas Paaso. A performance comparison of web service object marshalling and unmarshalling solutions. In *2011 IEEE World Congress on Services*, pages 122–129, Washington, DC, USA, 2011. IEEE.

[164] Java. The java tutorials, lesson:introduction to jaxb, 2023. Available at https://docs.oracle.com/javase/tutorial/jaxb/intro/. (Accessed on 17 May 2023).

[165] Sobhan Esmaeili. Re:how can i find a transmission time formula for wsn? https://www.researchgate.net/post/How_can_I_find_a_transmission_time_formula_for_WSN /5e3c8d933d48b73dc26ac563/citation/download. (Accessed on 4 May 2021).

[166] Jozef Wozniak. Mobility management solutions for current ip and future networks. *Telecommunication Systems*, 61(2):257–275, 2016.

[167] Bromberg Yerom, Valerie Issarny, and Pierre-Guillaume Raverdy. Interoperability of service discovery protocols: Transparent versus explicit approaches. In *Proceedings of the 15th IST Mobile & Wireless Communications Summit*, pages 1–5, Mykonos, 2006.

[168] Martin Peters, Christopher Brink, Martin Hirsch, and Sabine Sachweh. A client centric replication model for mobile environments based on restful resources. In *Proceedings of the Workshop on Posters and Demos Track PDT '11*, New York, NY, USA, 2011. Association for Computing Machinery.

[169] B. Zhang, Y. Li, and Y. Liang. Impact of packet size on performance of tcp traffic with small router buffers. In *Proceedings of the 2017 International Conference on Electronic Information Technology and Computer Engineering (EITCE 2017)*. EDP Sciences, 2017.

[170] Anca-Raluca Breje, Robert Gyorodi, Cornelia Gyorodi, Doina Zmaranda, and George Pecherle. Comparative study of data sending methods for xml and json models. *International Journal of Advanced Computer Science and Applications*, 9(12):257–275, 2018.

[171] Kamir Yusof and Man Mustafa. Efficiency of json approach for data extraction and query retrieval. *Indonesian Journal of Electrical Engineering and Computer Science*, 4(1):203–214, 2016.

[172] Wolf-Tilo Balke and Jorg Diederich. A quality- and cost-based selection model for multimedia service composition in mobile environments. In *Proceedings of the IEEE International Conference on Web Services*, pages 621—-628, USA, 2006. IEEE Computer Society.

[173] Surya Nepal Julian Jang-Jaccard. A survey of emerging threats in cyber security. *Journal of Computer and System Sciences*, 80(5):973–993, 2014.

[174] SiteLock. 3 ways to prevent a url redirect attack, 2023. Available at https://www.sitelock.com/blog/prevent-url-redirect-attacks/(Accessed on 30 June 2023).

[175] R. Mohana. A proposed soap model in ws-security to avoid rewriting attacks and ensuring secure conversation. *International Journal of Information Security and Privacy (IJISP)*, 12(1):74–88, 2008.

[176] Mushtaq Ali, Mohamad Fadli Zolkipli, Jasni Mohamad Zain, and Shahid Anwar. Mobile cloud computing with soap and rest web services. In *Proceedings of the 1st International Conference on Big Data and Cloud Computing (ICoBiC)*, pages 10–18, Kuching, Sarawak, Malaysia, 2017. Journal of Physics: Conference Series.

# Publications out of Thesis

## International Journals

1. **M. C. Mohammed Shameer**, P. P. Abdul Haleem and Yazik K. Puthenpediyakkal, A lightweight Data Exchange Format for Mobile Transactions, *International Journal of Computer Network and Information Security (IJCNIS)*, Vol.15 No 3, pp.47-64, 2023, DOI:10.5815/ijcnis.2023.03.04 (Scopus). Chapters Concerned : Chapter 2 and 3.
2. **Mohamed Shameer M C** and Abdul Haleem P. P., A study on the requirements of a transaction model in mobile environment, *International Journal of Computer Science and Information Technologies*, Vol. 13 (5), pp.103-109, 2022 (DOAJ, Google Scholar). Chapters Concerned : Chapter 2.

## Journals Communicated

1. M. C. Mohammed Shameer, P. P. Abdul Haleem, A Lightweight, Reliable, Disconnected and Distributed Message Transaction Model using LXML Format in Mobile Environment, Communicated to Sadhana - Academy Proceedings in Engineering Sciences (Scopus). Chapters Concerned : Chapter 2, 4, 5, 6 and 7.
2. Mohammed Shameer M C, Abdul Haleem P. P., A Layered Approach for Checking and Isolating Rewriting Attacks in LXML Messages, Communicated to Indonesian Journal of Electrical Engineering and Informatics (Scopus). Chapters Concerned : Chapter 2 and 8.

## Conference Papers Presented

1. Data Exchange Formats in Mobile Transactions: A Comparative Analysis, National Conference on Computational Intelligence Practices and Technology (ConCIPT 2020), Farook College (Autonomous), Sponsored by UGC, January 2020. Chapters Concerned : Chapter 2 and 3.
2. Demonstration and Study on E2EE using Firebase and AES algorithm, First National Conference on Computational Intelligence and Data Analytics (CIDA 19), February 2019, Dept. of Computer Science, University of Calicut. Chapters Concerned : Chapter 8.
3. PhenoTypic Data Analysis: Issues and Challenges, National Conference on Advances in Computing, Communication and Applications 16, Sullamussalam Science College Areacode, March 2016.

# Appendix

| | |
|---|---|
| ACK | Acknowledgement (a confirmation for successful receipt of data). |
| AES | Advanced Encryption Standard (AES) is a widely used symmetric encryption algorithm. |
| Binary Format | A method of encoding and representing data in a binary (machine readable) form. |
| Checksum | A value generated from the data that is used to detect errors that may be introduced during transmission or storage. It is primarily used to verify the integrity of data and to ensure that data is not corrupted. |
| Content Density | The amount of meaningful data or information contained within the document against the total size of the document expressed in percentage. |
| CRM | Customer relationship management (CRM) is a technology for managing organizational relationships and interactions with customers and potential customers. |
| Data Interchange | Data interchange refers to the exchange or transfer of data between different systems, applications, or devices with an intention of sharing information or enabling communication, integration and interoperability among the communicating parties. |
| Data Interchange Formats | Data interchange formats are standardized ways of representing and encoding data to facilitate data exchange, integration and interoperability between heterogeneous systems, applications, or platforms. |

| | |
|---|---|
| Data Object (DO) | Data Object (DO) is a class or a user defined data type with a number of data members corresponding to the number of distinct tags in the LXML document or block. |
| Deserialization | Deserialization is the process of reconstructing data or objects from the sequence of serialized byte streams. |
| Digital Signature | A cryptographic technique used to authenticate a digital document or message and to verify the authenticity and integrity of it. |
| Disconnectedness | The mode of operation in which the system or application or device may continue to operate even when it is not actively connected to the network or to the central system. |
| Distributed | The term is used to describe systems, processes, services or resources that are spread out or decentralized across multiple locations or nodes rather than concentrating at a single centralized location or node. |
| DAO | Data Access Object. They are patterns that provide data operations without exposing database details. |
| EPOCH | EPOCH is a reference for measuring time that represents the number of seconds elapsed since 00:00:00 UTC (January 1, 1970). |
| ERP | Enterprise resource planning (ERP) is a software suite used by an organization to automate and manage key core business processes including accounting, resource management etc. |
| Extensibility | A feature that allows users to define custom, user defined, self descriptive elements and data structures. |
| HTTP | Hypertext Transfer Protocol. It is an application layer protocol for transmitting hypermedia on the world wide web. |
| HTTPS | HTTP Secure. |
| Interface Gateway | A component in the middleware system responsible for LXML message handling. |

| | |
|---|---|
| JSON | JSON (JavaScript Object Notation) is a lightweight, human readable data interchange format. |
| Lightweight | Lightweight is characterized by its simplicity, conciseness and efficiency resulting in minimum complexity and processing overhead making it suitable for environments where performance, speed and reduced resource utilization are essential. |
| LXML | Lightweight XML, A less verbose and lightweight data interchange format derived from XML. |
| Mapping Rules | Set of rules or protocols that govern the mapping process. |
| Mapping/Binding | The process of assigning each LXML value to a data member in the DO. |
| Marshalling | The process of converting an object to a stream of data. |
| Middleware | The software component that acts as an intermediary between different applications, services, or components within a distributed computing environment for facilitating communication, data exchange, and interaction between these components at the same time abstracting the complexities of network communication and platform differences. |
| NAK | Negative Acknowledgement - indicates a failure in transmission. |
| Overhead | Overhead refers to the additional amount of resources, time, or data required by a system or communication process beyond the core task, introduced due to the workaround or change carried out. |
| Pagination (Segmentation) | Breaking down the LXML message into smaller units for transmission. |
| Parsing | Parsing is the process of reading a document and extracting its content. |
| Reassembly | Regenerating the original message after segmentation. |

| | |
|---|---|
| Reliable | The ability of a system to deliver data accurately and consistently at the destination end and ensuring data is received properly without any errors. |
| Resource Constrained Networks (RCNs) | Communication networks with devices that suffer significant limitations in availability of resources such as bandwidth, processing power, memory, battery power and energy. |
| REST | REST stands for Representational State Transfer. It is an architectural style for designing networked applications and web services. |
| Rewriting Attacks | Intentional insertion of fake or irrelevant data elements to the XML structure without changing the signature. |
| Schema | A formal specification that defines the structure, data types, constraints, and other rules for validating a document. |
| Schema aware | Conforms and satisfies the schema rules and constraints. |
| Serialization | The process of converting an object or data into a stream of bytes so that they can be easily stored and transmitted. |
| SOA | Service Oriented Architecture is a design approach for building software systems using loosely coupled and reusable components or services. |
| SOAP | Simple Object Access Protocol. It is a XML based protocol for accessing web services over HTTP. |
| Symmetric Encryption | An encryption standard that uses same secret key for both encryption and decryption. |
| Transaction | A sequence of one or more operations such as data retrieval, updates, or other tasks carried out by wireless devices or nodes that are treated as a single, atomic unit of work. |
| Transaction Model | To manage and coordinate transactions and other data exchanges among devices or nodes. |
| Transmission | The process of sending data or information from a device or location to another through a communication medium. |

| | |
|---|---|
| Unmarshalling | Reverse process of marshalling. Unmarshalling refers to the process of converting a stream of data back to objects. |
| Verbosity | The excessive use of descriptive elements, tags, attributes or other formatting information than actually necessary leading to larger file size and complexity. |
| Web Services | A collection of open protocols and standards used for XML based information exchange between diverse systems and applications. |
| X.509 | A standard format for public key certificates that enables secure communication and transaction between two parties. |
| XML | A markup language based on Standard Generalized Markup Language (SGML) used as a file format for storing and transmitting data between heterogeneous systems. |
| YAML | YAML Ain't Markup Language is a human readable, concise data serialization language. |