Building a Secured XML Real-Time Interactive Data Exchange Architecture

Yousef E. Rabadi

A thesis submitted to the University of Huddersfield in partial fulfilment of the requirements for the degree of Doctor of Philosophy

University Of Huddersfield

November 2011

Abstract

Nowadays, TCP and UDP communication protocols are the most widely used transport methods for carrying out XML data messages between different services. XML data security is always a big concern especially when using internet cloud. Common XML encryption techniques encrypt part of private sections of the XML file as an entire block of text and apply these techniques directly on them. Man-in-the-Middle and Cryptanalysts can generate statistical information, tap, sniff, hack, inject and abuse XML data messages. The purpose of this study is to introduce architecture of new approach of exchanging XML data files between different Services in order to minimize the risk of any alteration, data loss, data abuse, data misuse of XML critical business data information during transmission by implementing a vertical partitioning on XML files. Another aim is to create a virtual environment within internet cloud prior to data transmission in order to utilise the communication method and rise up the transmission performance along with resources utilisation and spreads the partitioned XML file (shredded) through several paths within multi agents that form a multipath virtual network.

Virtualisation in cloud network infrastructure to take advantage of its scalability, operational efficiency, and control of data flow are considered in this architecture. A customized UDP Protocol in addition to a pack of modules in RIDX adds a reliable (Lossless) and Multicast data transmission to all nodes in a virtual cloud network. A comparative study has been made to measure the performance of the Real-time Interactive Data Exchange system (RIDX) using RIDX UDP protocol against standard TCP protocol. Starting from 4 nodes up to 10 nodes in the domain, the results showed an enhanced performance using RIDX architecture over the standard TCP protocol.

Contents

List of Figures

List of Tables

Acknowledgement

Through Christ all things are possible. Therefore, I give thanks to God for giving me the opportunity and perseverance to complete this work.

To my wife, I would like to express my most sincere gratitude for encouraging and motivating me to complete this research and to my family members for their moral and continuous support. I would also like to thank my supervisor Dr. Joan Lu for her invaluable support and for helping me identifying the key areas of this research. Thank you to all supporters, the information you provided greatly contributed to achievement of this study.

Acronyms

ACL:	Access Control List
AMI	Amazon Machine Image
API:	Application Programming Interface
ASR:	Automatic Speech Recognition
AXML:	Assertion Markup Language
CA:	Certificate Authority
CORBA	Common Object Request Broker Architecture
CR:	Compression Ratio
CT:	Compression Time
DES:	Data Encryption Standard
DOM:	Document Object Model
DoS:	Denial of Service
DR:	Data Ratio
DT:	Decompression Time
DTD:	Data Type Definition
EFS:	Encryption File Systems
EC2	Elastic Compute Cloud
EXI:	Efficient XML Interchange
FIX	The Financial Information eXchange Protocol is a series of messaging specifications for the electronic communication of trade-related messages
HTTP:	Hypertext Transfer Protocol

IDL:	Interface Definition Language
IIOP	Internet Inter-ORB Protocol is a protocol that makes it possible for distributed programs written in different programming languages to communicate over the Internet
KLOC	A metric of one thousand lines of code used to measure the size of a software program
MOML:	Media Objects Markup Language
MRCP:	Media Resource Control Protocol
OASIS:	Organisation for the Advancement of Structured Information Standards
OSI	Open Systems Interconnection model is a product of the Open Systems Interconnection effort at the International Organisation for Standardisation. It is a prescription of characterising and standardising the functions of a communications system in terms of abstraction layers
PAP:	Policy Administration Point
PDP:	Policy Decision Point
PEP:	Policy Evaluation Point
PIP:	Policy Information Point
PKI:	public key infrastructure
PRK:	Private Key
PUK:	Public Key
RD:	Regular Documents
REST:	Representational State Transfer.
RMI	Remote Method Invocation part of Java RPC
RPC:	Remote Procedure Call
RTP:	Real-Time Transport Protocol

S3	Simple Storage Service
SAX:	Simple API for XML
SD:	Structural Document
SGML:	Standard Generalized Markup Language
SIP:	Session Initiation Protocol
SIT	Structural Index Tree, a tree that represents an XML file
SNMP:	Simple Network Management Protocol
SOA:	Service Oriented Architecture.
SOAP:	Simple Object Access Protocol.
TD:	Textual Document
UDDI:	Universal Description and Discovery Interface
UUID:	Universally Unique Identifier
W3C:	World Wide Web Consortium
WSDL:	Web Services Description Language
WSDL:	Web Services Description Language
WS-Policy:	Web Service Policy
XML:	Extensible Markup Language
XPath:	XML query language
XQuery:	XML query language
XSLT:	XML Style sheet Language Transformation
XWRT	XML Word Replacing Transform, a Lossless XML compressor

Copyright Statement

CHAPTER 1

1. Introduction

From the industrial age the world has become more in need to connect businesses together and globally migrate to an informational age, and exchanging information has increasingly become a part of its requirements. However, technology is used not only by ethical users but unfortunately by unethical users too; they grab any opportunity to manipulate or spy on the data during transmission for their immoral purposes. The need to set up an interoperable framework for exchanging data between different domains plays an important and essential role in doing business. As XML increasingly becomes a standard format for transmitting data on networks between different businesses, the need for finding secured and efficient techniques of transmitting XML data files is an essential matter. Real-time interactive data exchange system (RIDX) aims to place a multi-layer of defence of exchanging XML data information electronically between businesses, organisations, and other groups using Internet cloud network as a platform. The RIDX goal in this study is to build a clear infrastructure required for managing XML data real-time interactive exchange, taking into consideration the enormous security threats that face XML data files during transmission. The proposed system adapts as part of its transport protocol by using a multicast data distribution mechanism in order to reduce network resources, minimise the hindrance and expenditure. Thus far, there has been little stimulus to utilise multicast data distribution, because it lacks a protection mechanism for the data being delivered.

Although there has been significant progress in presenting secured multicast data distribution different drawbacks, threats and attacks can be addressed during the traditional

transmission process:

- Communication channels that are not secured will jeopardise data integrity, from modification, data misuse, and data abuse. A man-in-the-hub can tap the stream of data including encryption keys, sensitive data etc.
- Operating system security breaks including memory observation; invaders can forecast sensitive secrets concealed inside memory.
- To prevent a service from providing its ordinary functions that may result to a non-contemporary information, this denial of service (DoS) can cost the organisation a large amount of service time and money.
- When transmitting data across a shared network, some tools can be used to sniff network packets in order to reveal sensitive information especially if no data encryption measurements have been taken.
- When encryption methods are applied on sections of XML file assumable confidential parts, cryptanalysts can generate statistical information and extract some useful data that can be used unlawfully, and if the encryption is applied on XML as one block of data, efficiency problems might occur.
- Validating messages transmitted from the originator is always a concern. Therefore an authentication process as a security measurement has to be established.
- Failing to preserve data integrity from any alteration, maliciously or accidentally, is a challenge during data transmission.
- Failing to guarantee that the data transferred can stay confidential and convey assurances that the data is not revealed by any unauthorised parties.

1.1 CONTRIBUTION & IMPORTANCE OF THIS STUDY

A significant contribution of this study is to create a virtual multi agent decentralised domain network using Internet cloud. A group of nodes communicate and establish a ground for various communication protocols and interoperable XML data exchange among them. Process and embedded security measurement stages such as automation, authentication, encryption and tunnelling protocols are used to ensure communication security and service availability and to sustain service on demand prior to XML data transmission in order to conduct a secured XML data exchange between domain members, businesses and organisations. The aim is to build embedded multi-level security prevention and defence mechanism in order to transmit XML data files in a secured manner coupled with reliability, quality of communication, independence, scalability, and flexibility.

Another contribution for this system architecture is to implement multicast reliable communication to deliver XML data files in multi-concurrent connections to achieve the best transmission efficiency during multi-concurrent transmissions, which will widen the simultaneous number of connections to serve data transmissions and enhance transmission performance in terms of speed, message rate and throughput. Using a multicast reliable communication technique can take advantage of the availability of RIDX instances which reside as decentralised multiagents in the Internet cloud; this will allow the sender to elect the most efficient nodes including the utilised communication paths and routers that will carry out messages to different destinations.

A constructive contribution to this approach is by fragmenting vertically an XML data file into customised smaller chunks, to form a meaningless set of characters in each chunk in order to hide the structure of the XML file, and to conceal sensitive information that the XML

file might contain. Furthermore, apply encryption techniques on each chunk to add another layer of security on each partition, and then send the fragmented/shredded chunks of each piece into different paths through RIDX virtual multi agents' domain network which reside on Internet cloud, until all parts (chunks or messages) reach the final destination from all around network multi agents (domain nodes) and merge back into its original file structure through a set of modules. This method will utilise the network resources for better performance by avoiding a network bottleneck problem, and in addition, it will minimise the risk of tapping messages and analysing data messages, furthermore, distributing the load and breaking down the risk among its multi agents that form the domain group.

1.2 RESEARCH MOTIVATION

XML has come into view as the de facto preferable structure for saving and interchanging data through the Internet. While data is exchanged over the Web, security matters become progressively more essential. Such issues extend from one level to another: data security level to network security stage up to high level access controls; the author's focus is on how to create an environment which employs precautionary security steps in order to secure XML data transmission in the Internet cloud.

Developing a system that enhances both XML security and communication efficiency within Internet cloud architecture is a challenge. In cloud computing, different networks and various computers' software interact with each other, there are always security measurements that need to be considered and network efficiency needs to be enhanced.

By taking different threats into consideration when building a system, there is a belief that this new approach contains a mechanism that is apposite to achieve the security

requirements for various threats; a minimum requirement is at hand such as authentication, integration and confidentiality.

Adding multi-level stages of defence is the key to this project:

- Level 1: Create a multiagent virtual network domain group within Internet cloud and establish connectivity between all nodes in the domain prior to any data transmission, while enforcing methods of communication by ensuring authenticity, encryption and transmission efficiency using reliable multicast techniques.
- Level 2: Vertical data shredding of the XML file to create a data set of characters into data chunks to hide XML data structure and sensitive information to be ready for transmission and spread across the virtual domain multiagent network until they reach the ultimate destination side.
- Level 3: Adding another security layer by applying tunnelling and virtual private network techniques between domain members and initiating tunnelling techniques within the automation process of establishing secured connectivity.
- Level 4: An embedded and automated process flow and employing a communication set of rules such as node authentication and public/private key encryption, in addition to accessing the system by understandable API routines to govern the processes from initiation stage until the process termination without any human interference, and relatively lowering the complexity of establishing a combination of security measurements and lowering the operational cost.

1.3 RESEARCH HYPOTHESIS

This thesis scrutinises various XML security threats, especially during XML transmission. Being able to preserve security throughout XML transmission along with a high transmission performance are crucial elements of any XML transmission and XML security techniques.

Following on from the arguments developed in the previous section, the research hypothesis is:

1- Creating a virtual domain network in an Internet cloud prior to data transmission can increase system availability and enforce process automation.
2- Customising and adding functionality on standard UDP can enhance data transmission efficiency and utilise resources over standard TCP protocol.
3- A multipath multicast connectivity transmission can present better transmission performance in terms of speed, message rate and throughput rather than point-to-point connectivity transmission.
4- Applying embedded security measurements such as authentication, encryption, XML data vertical partitioning and tunnelling in a virtual network domain between members will enhance security against threats.

Below, a real-time interactive XML data exchange system is presented and evaluated to assess its features and the performance improvements, taking into consideration the enhancement of the automation of the security methods that has been used.

1.4 THESIS OVERVIEW

During a project led by the author to build financial application software that works as a middleware and set the integration between the stock market and brokerage houses, the middleware module has to meet certain requirements posed by the stock market authority committee in order to be able to connect to the stock market and make successful trading sessions. The stock market adopts a message specification structure presented by the Financial Information eXchange (FIX); they set a standard protocol for real-time electronic exchange, which consists of tag-value pairs separated by a special delimiter character (SOH, which is ASCII value 0x01). Afterwards, due to the demand of expanding FIX message protocol to adapt a wider range of applications standards, FIX organisation introduced a newer protocol by adopting XML structure for message representation called FIXML. During the project, a considerable amount of money was needed to invest in various equipment and different technologies in order to establish a successful communication to the stock market, for example, various hardware/software was required such as routers, firewalls, data management system and access control systems to implement the security measurements to establish a secured connectivity that matches the criteria according to the security standards set by the Market Authority committee. A dedicated team of network engineers to maintain the communication on a regular basis and with continuous supervision added to the cost, and furthermore, with a high cost of establishing leased lines dedicated for this purpose only, these requirements led to a high operational cost and manpower efforts.

The project motivated the author to carry out research and study alternatives for establishing a secured communication to be more cost-effective, less complicated, an automated

process, scalable, service availability, secured and high communication efficiency. Therefore, important questions have been raised that need to be investigated and need to be answered comprehensively: what is cloud computing and how can we benefit from it? What is the main web service used in Internet cloud that is using XML as an intermediate file structure? And what are the methods used to secure XML communication between servicers and how does it control the access? As an outcome of the study the author has alienated the subjects into five more chapters as follows:

In section 2.2, the author starts to investigate different aspects of using cloud computing as a platform, and judges the effectiveness of cloud calculating when utilised through an association. This section examines the benefits and detriments if it may be an executable option in addition to finding out different challenges and the advantages of using the Internet cloud. Then, in sections 2.3 to 2.7 the author comes across different XML security threats, XML transfer limitations, and different techniques of securing XML files, together with encryption, XML transformation and partitioning. In addition to discussing the most common and traditional methods used to encrypt XML files (such as DES) sections of a file are encrypted as an entire piece within an XML file, or XML file as a whole block. These methods have shown some drawbacks. Therefore, the vital idea in the transmutation is how to create encoded values based on the actual file. The fundamental plan is to initially partition vertically XML data into smaller chunks that represent a data set, in addition to encrypting units individually.

In section 2.8 the author discusses the concept of how SOAP permits both COM objects and Java objects to communicate with one another in a decentralised Web-based and distributed environment, and also discusses SOAP's functionality and the challenges that are facing this Web service along with its limitations, and the differences between SOAP, CORBA and SOAP-

RPC. Why SOAP is considered a middleware between object elements of different kinds, to be employed as a medium for communications is also discussed.

Sections 2.9 and 2.10 study different techniques of XML-based access controls using encryption as well as the signature of XML as specifications to offer the mechanisms to encrypt and sign XML documents in the critical scenario of electronic-services, and engages the application of crypto-keys, in addition to incorporating digital certificates and PKIs with applications that use XML structure.

A combination of useful ideas composed from chapter 2 to form novel schemes that shaped the RIDX system is described in two chapters (3 & 4). Chapter 3 describes the structure and architectural design of the RIDX system, the components, along with different modules and its functionality that is used to process XML data files between application layer to network transport layer and vice versa. Chapter 4 describes the RIDX transport layer, which consists of three main modules and a set of supporting modules to implement and support the fully functional transport layer. In addition, the author sheds some light on different threats that might affect the network OSI Model, and how it can add another security layer by implementing tunnelling techniques.

Chapter 5 focuses on case studies that measure RIDX system performance, by comparing the performance (message rate and throughput) between RIDX UDP and the standard TCP protocol, also comparing RIDX performance against other case studies, which represents a one-to-one communication performance in terms of throughput, message rate and resources needed to complete data transmission. The final chapter (6) concludes the thesis and gives a concise summary of the achievements and a set of conclusions along with future work recommendations.

CHAPTER 2

2. Background Research

2.1 RELATED WORK

One of the protocols and services that has been used to transmit XML data format is SOAP (Simple Object Access Protocol). Evaluation studies of SOAP XML transmission protocol have been made, specifically of the SOAP and XML performance, with several of them concluding that the performance of SOAP and XML acquire a considerable price when compared to binary transmission protocols.

An evaluation experiment was conducted about SOAP implementation latency performance, comparing CORBA/IIOP and Java RMI protocols. As a conclusion, SOAP is much slower, therefore, SOAP XML messages are not appropriate for transmitting bulk data. Another comparison has been made while SOAP did fare poorly when compared to both binary CDR and the established industry protocol FIX.

Another implementation of exchanging messages is the FIX protocol, which stands for “Financial Information eXchange”. This protocol was developed to form a standardisation protocol for exchanging transactions electronically for trading securities in a real-time manner within and between stock markets, and also formed as standard protocol between brokerage houses or investors with stock markets. FIX messages consist of tag-value pairs separated by a special delimiter character (SOH, which is ASCII value 0x01), messages in FIX protocol hold different types of values, such as integer, floating point values and strings. It is a fast and reliable protocol that guarantees data delivery, but this approach does not comply with XML structure,

and FIX protocol is purely for financial trades, stock markets and securities trades, and the purpose of it is not transmitting XML data files between different types of businesses.

Another implementation which is derived from FIX protocol, is FpML and FIXML protocols, which is also used for pure financial interchange trading systems, and does not solve the XML data file transmission between different applications or different types of businesses, furthermore, it depends on a predefined DTD structure of a message including tags and values, which means it is not designed for transmitting XML data files as an independent middleware interchange between other types of businesses.

Another approach is REST (representational state transfer), which is a Web service depend on method of Request-Respond over HTTP protocol (methods such as PUT, GET, POST etc) that transfers data in a form of XML document. Therefore, a limitation of over 4 KB of data makes GET versus POST impossible, and also impossible to encode such data of URI, which gives an error “HTTP Code 414 - Request-URI too long”.

2.2 CLOUD COMPUTING

Businesses have turned out to rely on the Internet for their advertising, supply chain, and sales. E-commerce has gone through a wonderful expansion in the last period. Associations are searching for novel ways to purchase the Internet for business growth. Web services is a rising technique in the Information Systems field. Information Management is also an ever-increasing endeavour for several associations. The requirement for detail is enhancing as they build up all the way through the detail age. Presently, several associations have static information architecture. They appraise the biggest requirement for information that the association possibly needs, add a security feature, and then buy and expand an information infrastructure focused on

that evaluation (Wilkinson, 2005). A review of the utilisation and architecture is carried out, and the benefits and drawbacks of procedure are surveyed. Various products are discussed as well.

2.2.1 Areas & Functions in Cloud Computing

Cloud computing can offer a variety of functions, from applications to data storage. IBM refers to cloud computing as “a term used to describe both a platform and a type of application”. Weinberg (2009) claims that cloud computing “is less a new technology than it is a way of using technology to achieve economies of scale and offer self-service resources that are available on demand”. The use of cloud computing to handle overflow requests is sometimes referred to as ‘cloud bursting’. This technology enables an organisation to handle and control infrastructure and applications while leveraging the cloud to expand and contract dynamically. Cloud bursting investigates two issues that many organisations entertain:

- An organisation periodically needs the additional capacity but return on investment for this capacity is long because of the sporadic use of the additional capacity.

- Companies are hesitant to move their entire infrastructure to a cloud due to security and reliability concerns.

InfoWorld provides seven areas where cloud computing can provide an impact:

1- Software as Service (Saas) - provides a single tool through a web browser. The consumer doesn’t demand straight servers, licences or investment.

2- Utility Computing - presents virtual servers and storage on demand that an organisation can access and use.

3- Web Services in the Cloud - instead of delivering full applications, this service allows users to access APIs for added services.

4- Platform as a Service - this offers an environment to develop as a service. This technology can be used to develop customised applications that dash on provider's server.

5- Managed Service Provider (MSP) - this is a service provided application that actually resides in the vendor's cloud. Examples are anti-virus and anti-spam applications.

6- Service Commerce Platforms - this type provides a service hub in order for the user to interact with. It is a mixture of SaaS and MSP. It can be thought of as an automated service bureau (Amer-Yahia, 2002).

7- Internet Integration - this service is in its infant stages. This will provide the ability to provide integrated solutions to its customers. The concept is that giant clouds could be melded together.

Cloud computing was anticipated to be one of the main Information Systems styles in 2009. Weinberg defines cloud computing among his Nine Hot Technologies for 2009 (2008). He describes three levels of clouds. The first level is infrastructure in the cloud. The customer is aware and informed about their infrastructure and they make selections about the type of infrastructure desired. The customer only pays for the resources used and can scale up or down as needed. The next level or layer is the web development platform. Google's App Engine for example allows web developers to develop and upload web-based code and rely on Google's infrastructure to install the application and allocate computer resources. The third level is running application enterprises in the cloud. A cloud vendor can take over an organisation's enterprise and manage the availability and performance of the enterprise. E-mail may be the

most likely and common enterprise to be managed by cloud computing. To get an insight into the features of cloud computing it is useful to investigate products that have emerged. Amazon, Google, and IBM all offer cloud computing services. Microsoft is beginning to roll out its own cloud computing endeavour (Wilkinson, 2005).

2.2.2 Amazon's Role in Cloud Computing

Amazon Web Services (AWS) emerged from the challenges and experiences that amazon.com has grown through. Amazon features two levels of cloud computing:

- **Raw computing resources**: Offers traditional resources such as processors, memory, file systems, and messaging. Amazon supplies and manages the resources and charges on a per-use basis.

- **Ready-to-go appliances**: Build applications or services on top of raw computing resources. Amazon has partners who generally provide the ready-to-go appliance. The partners charge a surcharge on top of Amazon's charge for the service. In this case, Amazon is purely an enabler; it depends on yourself or the partners to create the applications. Some vendor solutions manage the AWS resources directly.

Amazon's AWS consists of five services:

1- Elastic Compute Cloud (EC2). This product offers virtual equipment containing raw resources: storage, memory and processors. All of the virtual machines are based on Linux. Amazons with its various applications that work in cloud can allocate and deal with locating the virtual machines. EC2's virtual machines range from 32 bit to 64 bit, complete with associated memory and storage. EC2 is currently in beta stage and does not offer any service level

agreements (SLA). For security, a virtual machine can have a firewall attached and forms a security group. Also, a security group features protection to and from IP ports, IP protocol, or group. The Amazon Web Services site lists the following functionality:

- Produce an Amazon machine image (AMI) comprising libraries, applications, and data and connected configuration settings. Or use pre-configured, template pictures to get up and working instantly.

- Upload the AMI into Amazon S3. Amazon EC2 presents applications that keep securing the AMI simple. Amazon S3 presents a secure, fast and reliable repository to keep images.

- Apply Amazon EC2 web service to arrange network and security access.

- Select which instance styles and working system you wish, and then begin, terminate, and observe different examples of your AMI as required, applying the Web service APIs or the range of management tools presented.

- Decide whether you wish to perform in multiple areas, apply static IP endpoints, or connect constant block storage to your examples.

- Pay only for the resources that are used, like data transfer or instance-hours.

2- Simple Storage Service (S3) - This is a simple, direct, large storage service. A customer obtains buckets and places data objects into the buckets. A security profile controls access for the bucket. There are several options or added features available. Third-party vendors offer an application that offers a file system view of the buckets you have purchased. URLs for direct access to a bucket are also available. Specific functionality includes:

- Read, write and remove aims comprising from 1 byte to 5 gigabytes of data each. The number of aims that can be secured is unrestricted.

- Every aim is secured in a bucket and retrieved through an exclusive, developer-planned key.

- A bucket can be situated in America or the UK. All aims within the bucket will be kept in the location of the bucket, but the aims can be contacted from anywhere.

- Authentication systems are presented to assure that information is secured from unofficial contact. Aims can be prepared publicly or privately, and privileges can be given to particular consumers.

- Implements standards-based REST and SOAP conditions prepared to perform with any toolkit of Internet-development.

- Made to be flexible so that functional layers or protocol can simply be added. Protocol of default download is HTTP. A Bit Torrent protocol line is presented to lower costs for high-scale distribution.

- Consistency backed with the Amazon S3 Service Level Agreement.

3- Simple Queue Service (SQS) - provides a message queue. This service provides message storage and provides guaranteed message delivery. SQS is highly reliable and can store messages for later delivery if desired. It interconnects with all of the other AWS services.

4- Simple DB - provides a rudimentary query language interface for textual data. It is currently in a limited beta stage. Up to 256 attributes can be associated with your data and basic Boolean operations are used for queries (Muehlen, Nickerson and Swenson, 2005).

5- Cloud Front – delivers access to Amazon's worldwide distributed configuration of servers. Cloud Front integrates with S3, supplies high transfer rates and low latency. S3 contains the original data and relocates it to the necessary edge server. The price for this service varies with use (Abiteboul, 2001).

For access and activation, one must obtain an account. The account is accessed by use of an access key or security certificate. There are three interfaces that your applications can interact with:

Rest – assembles an HTTP request message containing data wrapped within a request main body.

Query – relies on HTTP but utilises basic browsers using names and parameters.

SOAP – uses XML documents described in a WSDL.

The AWS applications correlate to the interface methods as described in the table below:

	Rest	Query	SOAP
EC2	--	X	X
S3	X	--	X
SQS	X	X	X
Simple DB	--	X	X

Table 1: Interface correlation methods

There are also two other types of interfaces available that are useful to the user: the AWS toolkit and third party vendor products. Persistence when using AWS revolves around S3. When

one sets up an account they have the ability to create up to ten persistent storage buckets, with each being unique. The storage buckets can be associated with a URL if desired. A third party vendor can be used to view the storage as if it were a file system and also will provide the ability to back up your storage directly into the cloud (Muehlen, Nickerson and Swenson, 2005).

2.2.3 Google's Role in Cloud Computing

The Google Application Engine (GAE) relies on computing capabilities as for AWS it is more focused on computing resources. The GAE features application resources featuring an agile computer language (Python) and will shortly include the Java programming language. The application engine allows a customer to perform web applications utilising the GAE infrastructure (Wilkinson, 2005). The application promises that it is easy to use, easy to maintain, and scalable. GAE offers e-mail and collaboration applications, including video streaming. The applications can be tailored to the needs of the individual or corporation.

When using GAE applications files can be shared or tightly controlled. They can be accessed by a URL if desired. Documents and presentations can be imported and exported into the service as needed, providing flexibility for the customer (Abiteboul, 1997). In its messaging service, Google promises a cost benefit (the degree according to size) and the following services:

- E-mail accounts with 25 GB per user and search tools
- Instant messaging, voice and video chat
- Group calendars
- Mobile access
- Spam and virus filtering

- Web-based collaboration applications

With its collaboration applications, Google claims the following features:

- Essential collaboration applications. It offers a website, document and video applications.

- Continuous innovation. Google claims to be continuously upgrading the site to provide up-to-the-minute technology. It recently expanded its programming applications to include Java.

- Smoother information sharing. The application site will enhance current productivity suites, can be stored on the site or copied externally, and provides myriad coverage of file extensions and compatibilities.

- Worker mobility. The data can be accessed anywhere, from any computer.

- Information access control. Information can be declared as public for open file sharing or private when stringent security measures are needed.

- Enterprise class service. Google Apps offers phone support and a 99% uptime Service Level Agreement.

- Secure infrastructure. Google Apps claims security advantages for several reasons. First, its use eliminates the need for file copying to external devices (a huge source of external virus and worms) and attachments since files can be directly accessed. Also, Google Apps has a dedicated security team and has installed perimeter defences (firewalls and intrusion detection). In addition Google is built from the ground up with security as a priority (Whitemore, 2009). An application goes through routine security reviews from Google as it is being developed. Finally, Google fractures its data across many servers and is non comprehensible to humans. Data is replicated in multiple data centres for redundancy for back-

up assurance. Finally, Google builds with only necessary software components, and homogeneous server architecture ensures rapid and clear updates.

2.2.4 IBM's Role in Cloud Computing

IBM has also entered the cloud computing forum, but in a different capacity to Google or Amazon. They offer to set up cloud computing organisations, by providing infrastructure, hardware and software, and information management to a potential cloud vendor. They can also provide APIs for potential third party vendors.

IBM claims two major areas of cloud computing: cloud infrastructure and cloud services. These consist of four major features:

- Dynamic Infrastructure. IBM offers technology to build an infrastructure that is agile, quick and flexible. This allows an organisation to meet the demands of modern day business.

- Service Management. Software that provides management services built around visibility (to see the inner workings of business), control (to manage business), and automation (to optimise business).

- Information on Demand presents an important vision for unlocking the business worth of knowledge for aggressive benefit through allowing corporations to make and influence confirmed data to maximise performance of business.

- Service Oriented architecture is a business-centric IT system that encourages incorporating your business as connected, repeatable, business services or tasks.

2.2.5 Microsoft's Role in Cloud Computing

Microsoft has entered the cloud computing fray by offering Windows Azure. They are venturing into the Web services environment offering known development platforms in an Internet-based environment. Windows Azure offers a development platform and applications for customer use.

The developer features eliminate the need for up-front purchase of software. They offer development in the Visual Studio and .NET platforms. Currently programming can be done in languages supported by .NET, and Azure plans to add more languages. Azure provides an on-demand compute and storage environment that is automated and configured for dynamic scaling to match the needs of the customer with a pay-as-you-go option provided.

Microsoft also provides applications that are ready for use by its customers. These include database and storage applications. They also offer an application and service that allows collaboration.

It would be beneficial to discuss the implementation of a cloud computing architecture. At the physical bottom layer the hardware is virtualised to present a flexible and agile platform that can present maximum implementation of resources. The next two layers represent the key to cloud computing infrastructure: management and virtual environment. These two layers ensure that the data centre is effectively managed and that its resources are properly configured and allocated (Whitemore, 2009).

2.2.6 Disadvantages of Cloud Computing

There are some risks with cloud computing and there are several controversies surrounding its use. Ashford cautions that cloud computing does not work in every context (2008). He stresses that many legal complications could arise with the use of cloud computing, especially in a highly regulated medium such as financial services. The Data Protection Act requires businesses to control how personal data is used and stored but this can be difficult when cloud computing is employed. Businesses must only select cloud computing services that allow them to avoid risk entirely or manage it at a reasonable level.

Another issue surrounding cloud computing is that it is still a new technology and is still an immature process. Most companies that are offering cloud computing services are offering short term gain but have no plan for long term sustainability. Companies can save as much as 40% on customer relationship models by employing cloud computing but these services generally do not include aids to improve processes over time (Grötschel, Alevras and Wessäly, 1998). Being able to deploy an enterprise level application quickly at a low cost does not always mean success.

Cloud computing is especially attractive during economic hardship. Low costs, fast deployment, and the lack of maintenance concerns are a strong pull for many companies. Some estimates claim that up to 60% of businesses intend to employ some sort of cloud computing. These factors may pull many companies who are not a good fit for this service into employing it by the promise of short term business savings without realising its long term implications. A full study on the short and long term benefits of cloud computing needs to be conducted to see if it is

useful for an organisation, keeping in mind that there are also legal implications (Grötschel, Alevras and Wessäly, 1998).

Cloud computing has many benefits, it comes at a price. The following lists five concerns with cloud computing:

1- Security. One of the advantages of cloud computing is that a client company does not have to worry about maintenance of the systems in the cloud. At the same time, security of the data falls to the provider of the cloud services. Data privacy, therefore, is out of the client company's control. They cannot directly guarantee the security of these systems. For some companies that demand a high level of security, even though a cloud provider may promise a high quality, secure service, the risk from the use of cloud computing is too great to utilise this service (Ioannidis, Keromytis and Smith, 2000).

2- Reliability. Reliability is a worry faced by cloud computing customers. Organisations not only require dependable applications, but also services as well. E-mail and messaging have to be delivered on a consistent basis. As an example, in the summer of 2008 Google's e-mail and Apps services were out of commission, and on July 20, Amazon S3 was out of service for eight hours (Bushell-Embling, 2010).

3- Privacy. Privacy is a main concern in all aspects of information technology, and it is no different in cloud computing. The major issue is that the customer does not manage their data. Another concern is that the cloud computing company itself may outsource part of its storage capacity so that the customer does not even know who is managing the data (Curbera and Duftler et al, 2002).

4- The Cost of Failure. If a failure occurs in a cloud, either through system breakdown or security breach, all of the data and services maintained in the cloud will be at risk. As more organisations join a cloud the cost of failure increases dramatically. If a large super cloud is created, it is possible that a catastrophic failure could cause devastating losses. It could be damaging enough to put a cloud service out of business.

5- Cost Versus Risk. Although the possibility of financial loss is mentioned above, many organisations are joining cloud services because they promise short term gains, or the costs outweigh the risks (Bushell-Embling, 2010). Many of the cloud companies are still new and the technology is in its infant stages so reliability will be an issue for some time, until the technology matures. An organisation needs to ensure that the savings gained by a cloud computing service will overcome the risks associated with using it.

2.2.7 Advantages and Benefits of Cloud Computing

There are a myriad of advantages to using cloud computing. With cloud computing, an organisation or individual can take advantage of a potentially endless array of processors, storage and bandwidth, and also a vast array of software and application services. The concept behind cloud computing is pay-as-you go: an organisation only pays for the utilities that they use. This is a big advantage for an organisation that needs flexibility in certain aspects of their information systems. One example is an organisation that has a peak demand for data storage, perhaps a spike in storage needs for a few months for every year. This organisation can purchase the extra storage needed for its peak period through a cloud, or it can purchase all of its storage through the same organisation and allow their data storage to dynamically expand and contract as needed, paying only for what is used.

Elasticity is a main selling point of cloud computing. It can expand and contract to add or drop resources as needed, purchase resources it does not have (from a vendor) and allocate resources to assure efficient storage. It allows scalability up or down without a huge cost increase. Another major attraction is the ability to have direct connections to software and applications. The cloud can hold more than your software. It can host its own databases and software as well as third party vendors' products as well. The direct connection provides high performance and avoids latency issues that can plague a distributed Web service (Comer, 2006).

Study notes benefits for both individuals and businesses. Personal benefits to using cloud computing include:

- Ease of use. 51% of personnel who use cloud computing note that ease and convenience are the major reason for their choice. Cloud computing eliminates the need for setup, anti-virus checks, etc.

- Accessibility. Cloud computing allows software developers the ability to design applications without the concern for hardware constraints. Any computer may be used as long as it is connected to the Internet.

- Information sharing. Files can be created, uploaded and shared by end users. Data files can be spread widely across the Internet and can be easily accessed. Cloud computing is more reliable and faster than information sharing through e-mail. Collaboration allows consistency across a large number of users, a change by one user is instantly available to all other users of the file.

- Low risk of losing data. Total systems failures of home computers happen frequently; at a much higher rate than a large corporation. The major reason for this is the organisations have a

more robust virus and attack system, and have in place a hardware and software management system that is managed by professionals.

Businesses have the following benefits when utilising cloud computing:

- Fast start up of companies and projects. Organisations that run a multitude of businesses can reap large benefits by using cloud computing. Most SaaS in the clouds are designed to enlarge capabilities or capacity instantly, with no increase in infrastructure, or purchasing any new licensing for software or even the need for any training of new personnel (Rabhi and Benatallah, 2002). Organisations can expand and grow without the burden of taking on new worries; new projects can be opened efficiently. This will increase profitability of many companies, and ensure professional support from new technologies they are trying to utilise.

- Comparatively small cost. Servers, software, hardware, licensing, and training can amount to a high cost. For small to mid-sized organisations this cost can be excessive. Cloud computing organisations allow a customer to rent vast resources at a comparatively low cost with a vast array of plans and options to choose from.

- Flexibility. In the current paradigm, an organisation must develop its own network and data management system. The process of development is lengthy, generally features cost overruns and schedule delays. If outsourcing is used, the process is controlled by an outsider and the organisation is dependent on their end product. Contracts for vendors must be written, and management must still track the progress of the project. With cloud computing the process becomes much easier and quicker. An organisation can switch applications and user software, buy or delete resources, and use new technologies without a long time lag needed in the

development process. A variety of applications and software are instantly available on the Web (Dan, 2011).

- Maintainability. Computer systems maintenance is expensive and cumbersome. Organisations allocate a significant budget (sometimes in the hundreds of thousands to million dollars) for information system maintenance. A significant cost saving can be incurred if the organisation utilises a cloud computing system. Responsibilities of upgrades and replacing faulty hardware, etc fall to the cloud provider instead of the organisation (Rabhi and Benatallah, 2002).

- Safety. Protecting sensitive information becomes more difficult and complex every day. Many organisations do not have the resources to build the security measures needed to ensure information privacy and data integrity. Some organisations may not be aware of the urgency of the need for information security. Cloud computing organisations can provide many of these services (Rabhi and Benatallah, 2002).

2.2.8 Summary of Cloud Computing

Cloud computing is one of the hot topics in information systems today. One survey of 1,300 software buyers disclosed that 11% planned on deploying cloud computing. Another survey of business technology professionals disclosed that 38% of their organisations are already using cloud computing or are seriously considering it. It is apparent that cloud computing is becoming more than a fantasy story; it is a reality and is rising as a viable force in the Information Systems profession. One Information Technology writer estimates that cloud computing grew in excess of 20% in 2009 (Dan, 2011).

Many advantages and disadvantages in the deployment of cloud computing have been discussed. It is clear that cloud computing can be a great benefit to many organisations, but it is

also evident that the organisation must have a need and a long term plan for its use. Deployment of these solutions must be made after careful consideration of the impact that this service can have on the organisation, both positive and negative.

Currie (2011) contends that as organisations move to the cloud their solutions will become more complicated in some respects and less complicated in others. Information systems will become more complicated in that technology is likely to be spread out or distributed across many vendors. Managing and controlling security will become a challenging operation for these organisations. Project management of all these solutions can be an equal challenge. The less complicated features for these organisations are that the working and functional mechanisms of hardware and software are moved off campus. These functions will not have to be managed.

The key to the successful use of cloud computing is planning. When using cloud architecture, assume things will fail. Design and deploy features from the cloud assuming failure will occur. Assume and plan for hardware failure. Four strategies can help plan for difficulty:

1- Plan for consistent back-up and restore.

2- Put up a procedure that will resume when rebooting.

3- Permit the condition of the system to re-sync the messages from queues.

4- Maintain preoptimised and preconfigured data images to sustain the reboot and re-launch.

Cloud computing appears to be a permanent fixture in the Information Systems world. It can provide cost savings, flexibility, and provide applications and data storage services to organisations that could not afford or manage them. It can be a wonderful and versatile tool if it is used and deployed properly. But, it comes with a risk, and with any associated business risks, mitigations and proper risk management must be employed if the deployment is to be successful.

2.3 XML SECURITY THREATS

Security is a critical issue (arguably the most critical one) for Web applications. Not surprisingly, there is a lot of activity around XML, Web services and security. The W3C promotes standardisation efforts for security, e.g. for XML encryption, XML signature or cryptographic keys. The security assertion mark up language is an XML-based framework promoted by the Oasis consortium for exchanging security information. The next example illustrates how basic security features may be supported in SAML. Suppose for instance that an SAML peer, Joe, wants to send a message to an SAML peer, Linda, with a portion of the message encrypted. In SAML, the portion of the message to be encrypted is a sub-tree rooted at some particular node, say n. To encrypt it, Joe has to remove the children sub-trees of n, say t1 to tm, and to replace them with their encrypted value.

In an SAML setting, encrypted XML data will be represented using the standard XML encryption with the following syntax:

<Encrypted Data Id? Type? Mime Type? Encoding?>
<Encryption Method/>?
<ds: Key Info>
<Encrypted Key>?
<Agreement Method>?
<ds: Key Name>?
<ds: Retrieval Method>?
<ds :*>?
</ds: Key Info>?
<Cipher Data>
<Cipher Value>?
<Cipher Reference URI?>?
</Cipher Data>
<Encryption Properties>?
</Encrypted Data>

SAML rely on a public key encryption scheme. Each participant has a unique public/private key pair denoted PUK/PRK. As usual, the private key is private, while the public

one is made accessible to the world, through the following service:

Public Key@ peer () -> string

A similar Private Key@ peer () service exists, that can only be invoked by the peer itself. It is assumable that each peer has the following generic services, which respectively perform encryption and decryption.

A main goal is to show how SAML provides a uniform framework for addressing standard query processing issues as well as issues such as security for data management, that are typically considered separately. The author believes that, in a Web context where various functionalities may be supported by different peers, it is important to provide an abstract model for distributed data management that captures these various viewpoints in a unique framework. Such a uniform model allows addressing issues such as data exchange protocol verification in a rigorous manner.

A Complete Framework

A whole threat-protection framework requires covering three key operations: prevention, protection and screening.

2.3.1 Prevention

A prevention framework should make sure that during the period of messages transmission to prevent through blocking possible message-level feats similar to the placing of attacks into the message. Message contracting, succession amounts, and the utilisation of PKI (public key infrastructure) among clients and services assists make sure to content and offers particular security next to man-in-the-middle and rematch assaults.

2.3.2 Protection

Software or infrastructure should be capable to defend it and downstream methods next to approaches that are planned to deliver it terminal. Long-familiar Web space assaults for example DDoS, payload poisoning, and exterior commands are as great a menace in XML and Web services usage. A well-designed treating architecture mixed with important safeguards may assist defends next to approaches.

2.3.3 Screening

Message-level testing must cover every traditional firewall and operates also as permission to the supervisor to permit or deny particular actions or messages. These operations comprise inclusive schema sustention, integrity enforcement, decryption/encryption; message capacity questions, biometric authentication, and some other permit or refuse criteria. The capability to assign or unload particular payload dispensation to some other best-in-class methods - for example a virus scan locomotive - permits safety controllers to tailor the assessment of the screening as needed.

2.4 DISCUSSION OF GENERIC COMPRESSION TECHNIQUES

XML is recognised mainly as a helpful and significant technique that was issued as an outcome of the huge reputation of the World Wide Web and HTML. Because of the effortlessness of its principle ideas and fundamental premises, XML has been utilised in figuring out several troubles, for example offering neutral information among wholly varied architectures, linking over the space among software systems with negligible attempt and putting in large amounts of semi-structured information. Nevertheless, this self-describing factor offers an enormous suppleness except conversely; it also brings in the important factor of verboseness of

XML documents which have outcomes in vast sizes of documents. As XML practice is growing, the enormous demand for competent XML compaction equipments have been subsisted. There are various research attempts to improve compression techniques. The utilisation of compaction equipments have several benefits, for example: diminishing the connection bandwidth needed to exchange data, diluting the storage capacity needed to minimise the storage and memory for query to process XML documents (Ho, Vincent and Cheng, 2009).

XML is a text agency for tree prearranged information. Therefore, an above board proportional access for compacting XML documents is to utilisation of the conventional all-purpose text solidity tools. Several algorithms were developed so far to expeditiously squeeze text information. The mainly famous also competent examples are: PPM, gzip, and bzip2 compactions. The gzip compaction is focused on the lossless DEFLATE information density algorithm that utilises the mixture of the LZ77 algorithm and Huffman coding. LZ77 algorithm gets compaction through swapping parts of the information with mentions to corresponding information that has previously moved during each decoder and encoder. Huffman coding employs an important technique for selecting the delegacy for every sign where the mainly ordinary qualities employ smaller strings of bits which are employed for fewer ordinary signs.

bzip2 compaction utilises the Burrows-Wheeler transmute for changing often chronic character series into equal letters draw, and after that attempts a move-to-front transmute, lastly doing the Huffman coding. Adaptive statistical information firmness method (PPM) focused on framework of the context modelling along with anticipation. It employs a set of statistical modelling methods, which may be deemed blending jointly various contexts in fixed-order to forecast next character in the input string. Forecast possibilities in every context in the model are computed as of occurrence calculates that are efficient adaptively. Even though PPM is merely

effective of the compactions obtainable, thus far, it is as well computationally the mainly high-operational priced.

2.5 XML NON-QUERIABLE COMPACTIONS

Suciu and Liefke (2000) demonstrated the initial execution of an XML witting compaction. XMill has brought in few novel plans for XML witting firmness which are pursued through various XML compactions. The important and most significant plan emphasised, is by dividing the XML structure as of information and the division of the information assesses into containers that are homogenously focused on the associative paths of these containers in a form of tree according to their information kinds. The XMill method does the compaction for XML files individually, one for the structure and one for the data values. XML tags and the features are put in the structure section and encoded in a dictionary-based style previous to allowing it to a back-end general text compaction method. XMill allots every separate part and assigns an integer code, this name/code provides the key within the attribute and element name dictionaries. In the information section, information assesses are sorted into semantically and homogenous associated containers in accordance to the path type and information.

Then every container is packed in individually employing specified compaction, which is suitable to the information kind of the container and the grouping procedure provides the restriction of the repetitions and thus raises the degree of compaction. The newest versions of the XMill spring dispersion, the compacted format of the intermediary binaries, may use one of three compaction methods (gzip, PPM, bzip2), which is a substitute of the back-end general intention compactions. Swacha and Skibinski have used the Word Replacing Transmute compaction that attempts a similar plan to XMill. It employs a dictionary-based solidity method (XWRT)

(Swacha and Skibinski, 2007). The plan of the XWRT method is to return commonly happening words with orientations into a dictionary that is sustained through a preliminary go through by above the information. Encoded outcomes presented by XWRT of the pre-working action to three substitutes common reason compaction schemes: PPM, gzip and LZMA. Figure 1 illustrates the process using XMill compression methods:

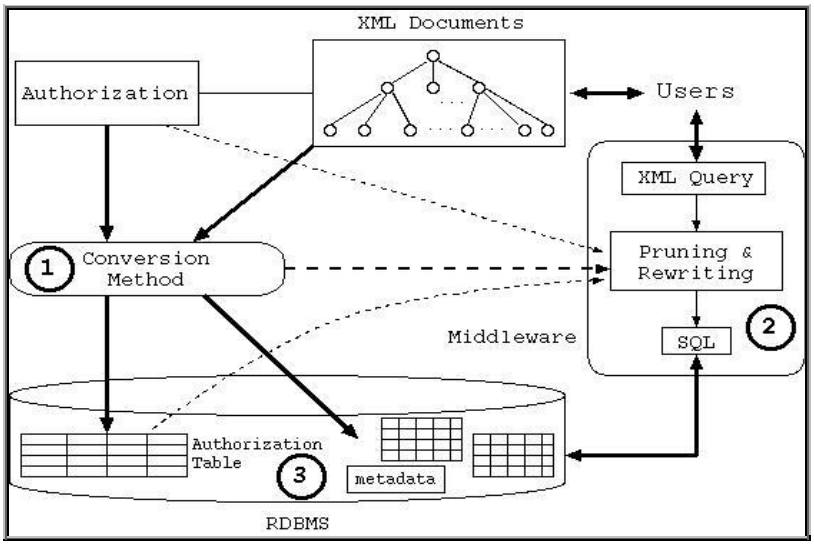

Figure 1: XMILL compaction (Suciu and Liefke, 2000)

Weimin Li (2003) has illustrated the operation of some other XML compaction that attempts the same plan of XMill with a trivial alteration. For example with XComp, the XML file is analysed initially, the constituents are re-arranged and sent on to the density engine. The analysed stage detail is utilised to limit containers' size in the memory. XComp exacted the effectiveness of the usage of the memory, which may be enhanced through attempting a storage window when setting a value.

2.6 XML AND ENCRYPTION

The World Wide Web Consortium (W3C) proposed several techniques to encrypt XML data, especially techniques that encrypt parts of XML files; their methods are flexible, thus even when encrypting parts of the document, but still preserving the structure of the XML file, therefore, this file can be encrypted many times from different parties also on various sections. In Ho Lam and James' point of view (2009), these approaches have some drawbacks, because W3C concentrated on the flexibility side of the method but not the efficiency of queries or the level of security, therefore, it does not gratify the needs of several applications where information safety and query effectiveness is imperative.

In Figure 2 below, the example covers XML fragment, the plain text segment as of '<Credit Card Id>' to '</Credit Card Id>' to be encrypted and restored by the encrypted information node. But this encrypted data is yet processed as a complete text block, at the same time interior structure is dismissed. A problem may occur where the redundancy inserted through the format of the XML document may be browbeaten to assault the encryption. For example, the encrypted section constantly terminates with the tag '</Credit Card Id>', and this may employ and be vulnerable to the cryptanalysis techniques. Another concern may be noticed especially when the secret section is returned through its representing blocks of encrypted data, the circumstances surrounding it possibly will be oppressed through the antagonist. For example, it may be noticed that the Credit Card ID for the encrypted block returns to John Rabadi. In addition, the facts that John Rabadi has some sensitive data (in our example the credit card ID) as displayed in the information file. Another case may be noticed that when numbers of purchased items are encrypted jointly, cryptanalysts could discover number of items by estimating the

length of the encrypted text block, therefore, they can generate statistical studies about it. In addition to these safety drawbacks, using this technique of encrypting data will create efficiency drawbacks. For example, the XPath query: //Payment Info [//Issuer = "Bank of Scotland"]/Name

Along with details of Payment Info, just detail regarding issuers is important to retrieve this question. Although the data that has been encrypted forms a whole block, it is not possible to extract the issuer information alone without bringing the whole block or maybe several blocks together. Accordingly, a prominent total of needless decryption is executed, which will eventually reduce speed query and exhaust the bandwidth.

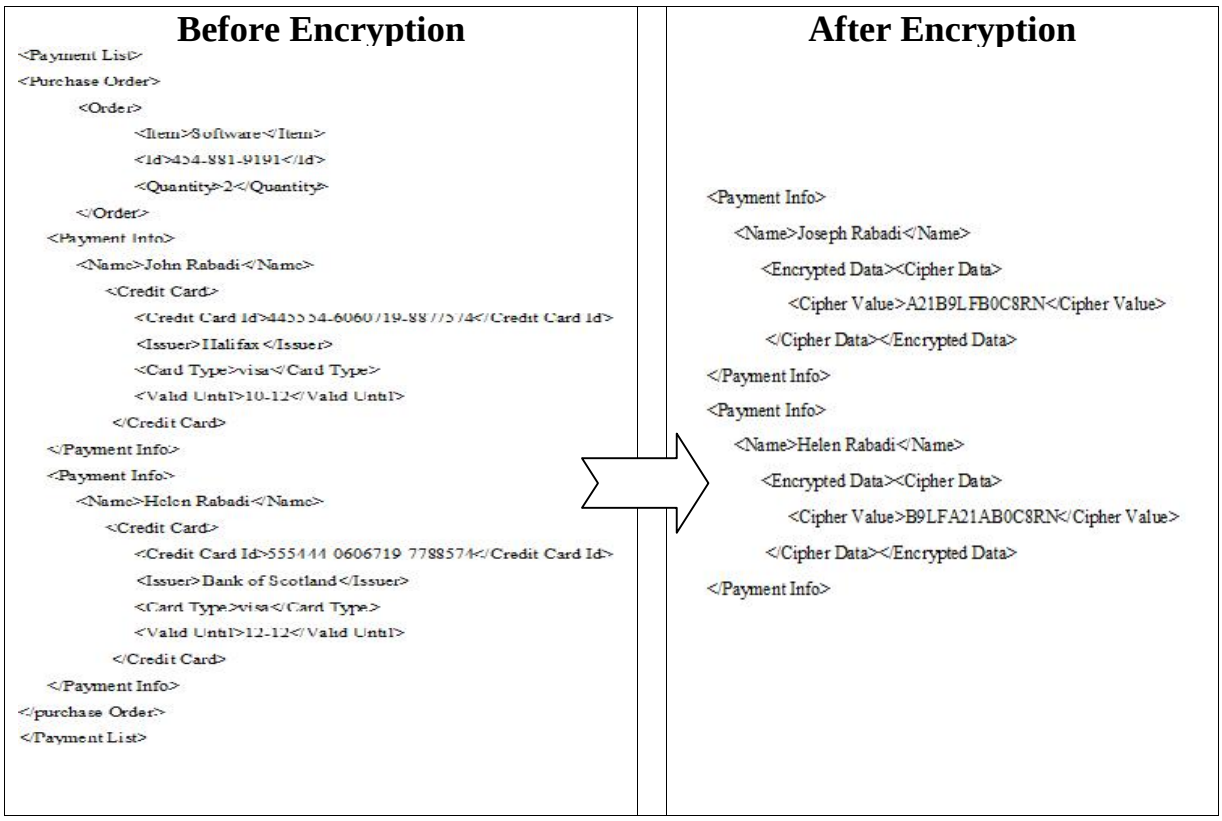

Figure 2: XML file before & after encryption

By considering both the effectiveness and the drawbacks, Ng and Yang (2006) proposed a new XML encryption technique (XQEnc); they focused on the expansions of XML repositories, which is the compaction using vectorization and skeleton methods. They

demonstrate the results of their method with other existing techniques; XQEnc showed an efficient ability in processing in the query retrieval and XQEnc method attempts to combine the security enhancement and query efficiency along with flexibility.

The foremost idea of XQEnc is to make XML data outsourcing secure. Safe XML outsourcing information makes it doable to save XML data in undependable servers and save XML information in a secured manner, moreover, reallocating the workload of querying the XML data on the server side as feasibly as it can, at the same time, making sure that the data is not exposed to the server outsourcing providers. The argument to outsource the server storage from a business standpoint is to take advantage of the service providers to manage the data in terms of back-ups, restores, archiving, managing the data efficiently etc, while the organisation can focus on the core of their business.

2.7 SKELETON & VECTORIZATION XML COMPACTION

In the XQEnc technique, the two Skeleton & Vectorization methods have been used; the latter take a broad view of ‘Vertical Partitioning’ technique, where the purpose is to optimise the query performance, especially in a relational database (Yang et al, 2006). An uttermost shape of vertical partition is to save every column separately in relational tables. So the main purpose of vectorization is to divide the XML document into vector paths. The outcome of this division is a series of information assesses happening beneath every path and posturing the similar path labelling.

2.7.1 Skeleton Compaction

Some other significant technology named ‘skeleton compaction’ was initially suggested for sustaining query working of compacted XML data. The purpose of this is to discard the severance included in the tree that has been formed for the XML file, through partaking an ordinary sub-tree and swapping consecutive and matching sub-trees (branches) with one branch and a diversity annotation. In the XML example above (Figure 2), it is noticed that the two (<Payment Info>) stubs after running skeleton and encryption methods are packed into one branch and one diversity notation. The results showed that the pressed XML skeleton is small and sufficient to fit substantially in the memory.

2.7.2 Vectorization

Vectorization is the technique of dividing the XML file into two divisions, one represents the structure of XML, and the other division is the data information values. It divides them into containers that are homogenously focused on the associative paths of these containers in a form of tree according to their information kinds. XML tags and the features are put in the structure section and encoded in a dictionary-based style, then the compression techniques allot every separate part and assign and integer code. This name/code provides the key within the attribute and element name dictionaries. In the information section, information assesses are sorted into semantically and homogenously associated containers in accordance with the path type and information.

2.7.3 Securing XML Outsource Data

Lately, the main concern of outsourcing and securing sensitive data has strained substantial consideration. In the outsourcing concept, customers rent a data storage server from

services providers to store their data in, but as organisations hold sensitive information about their clients, they need to secure their data when outsourcing it with a service provider, therefore, even when querying obtaining certain information from the server, there should be a way to secure the data within the server and to secure the requested data, also to keep the efficiency of the query high, therefore, the data should be stored in an encrypted form. In the meantime the server requires few details regarding the information (for instance the utilisation of 'crypto-index') so as to process questions. The outcome of an inquiry is typically an encrypted superset of the real outcome and is moved to the clients. The aim is to move query treating as far as possible to the server slope whereas sustaining information safety throughout information transfer and working.

A simplified architecture of information outsourcing method is an overview of the information outsourcing method as follows: a query from a user interpreted by the translator and dividing the query into two subordinate-queries. The first part of the query goes over the encrypted information, which is implemented at the server part through the assistance of a cryptograph index. The second part of the query filters the inquiry in the client part, the outcomes from the first part come up to the client and then chooses the actual answer. In order to do filtration, the data must first decrypt knowing that the outcomes from the server are in a form such as tuples. Various propositions suggested a way of forming the cryptography index; the purpose is to give helpful information to the server to progress the inquiries. The main suggestion is to initially divide name spaces into displaced containers, and then save the container IDs in a server repository.

While the query is processed, assessments are made to the inquiries and are interpreted into their representations of the ID of the container. This technique exposes the container IDs

from the actual saved information, and then the server retrieves a super set of the original outcomes. An extra effective method utilising the order conserves algorithm that ensures no detail outflow along with the optimal communication traffic. Most similar methods used to retrieve XML data have the retrieval methods only in the relational structure, and so far are not appropriate in XML structure (Yang et al 2006). Consequently basically replacing values through cryptographic indexes possibly exposes the detailed structure to the service provider or the host. In their campaigning instance, keeping sensitive information such as bank details or credit card information along with only assesses encrypted reveals the interior structure of the credit card chunk.

2.7.4 Overview of XML Query Encryption

The traditional methods used to encrypt XML files (such as DES) encrypt the file as an entire piece, but these methods have some drawbacks. Therefore, the vital idea in the transmutation is how to create encoded values based on the actual file. The fundamental plan of XQEnc is to initially calculate the compacted skeleton and the consequent vector that represent the data set, in addition to encrypting both units individually. While implementing XQEnc, the latter takes on the advance of using skeleton compaction plus vectorization to create the structural index tree (SIT). SIT assists in discarding the redundant and the replica formation in an XML file. In effect, prominent sections of the formation of several XML files are surplus and may be terminated (Atay, 2007).

By executing these methods, it avoids a complete decryption by grouping the information into several small blocks (chunks). Then utilising the algorithm for encrypting the data (similar to triple DES algorithm) for encrypting every part that is obstructed in XQEnc. Afterward, the

inscribed obstructs are joined to create the encoded values for the original file. To process queries in XQEnc, it needs initially to decrypt the related encrypted obstruct in order to reply back to the query.

2.7.5 Overview of XQEnc

As an XML file has an assorted nature, in several issues only sections of the original XML file require encryption, and these sensitive data sections possibly dispel over and are done without or with obvious patterns. A non-proportional way is to employ the techniques defined in the former part on every sensitive section of the file individually, that fulfil the W3C standards. Nonetheless, there are various downsides along with processing this approach (Miklau, 2006). For instance, if they utilise the XQEnc method on the payment info sections (see Figure 2), the two obstructs detail individually, and substitute them with their equivalent encoded text, the ensuing XML file is similar to the transformed file, excluding that the encoded text assesses are created by utilising XQEnc. Then the safety related issues remain that the content may still be victimised to assault the encrypted data and deduce analytical detail. Furthermore, the sensitive sections have precisely the same interior formation and the similar compacted skeletons are carried out several times. For now, the information vectors for a single sensitive section are frequently not big enough to occupy an information block, which badly affects storage use and inquiry effectiveness. Instead of encrypting every sensitive section separately, XQEnc places them jointly and creates only one part of encoded data, introduced as the very last position of the root tree. Utilising their campaigning instance, XQEnc creates the outcome as demonstrated in the following transmuted document:

<Payment Info List>

<Payment Info>
<Name>John Rabadi<Name/>
</Payment Info>
<Payment Info>
<Name>Helen Rabadi<Name/>
</Payment Info>
<Encrypted Information><Cipher Information>
<Cipher Value>E7FDA243B745CC586</Cipher Value>
</Cipher Information></Encrypted Information>
</Payment Info List>

As a result of the process, it derives two components for the encoded text: the compacted skeleton of the file (F), and the sensitive information shredded in vectors, the two components are encoded. Maintaining the compacted skeleton file (F) makes sure that there is no structural loss. The compacted skeleton file is generally not large, with a comparison between the original and the compacted one, is less than 1 MB for a file that extends several megabytes, or even a lesser amount of 1% of the total file size (Jianyong and Cunying, 2011). Therefore, the required memory is generally available in most devices nowadays, especially the tremendous enhancement of RAM size. Moreover, the option is still available by keeping just the 'partial' compacted skeleton that is applicable to the sensitive information. By doing so it will even create a slighter encoded value, but definitely the expenditure of processing the query is higher.

The sensitive data are kept in the shredded vectors along with their file position and the encoded value. Then to reply when XPath query is received, initially the compacted skeleton of the file is decrypted (Jianyong and Cunying, 2011). The XQEnc method is fired to process the

decrypted data from the previous step and treats both the values and the tags as two separate entity resources. Any non-sensitive values needed by the processor are retrieved directly from file. XQEnc shreds the file into chunks, and encodes each chunk separately. The smallest unit that is retrieved by the method is the chunk (Block). In their execution the non-encoded sections are initially analysed throughout the pre-processing stage, and evaluation of a non-encoded node may be simply fetched. If any details included in the encoded text node are required, the process of retrieving the encrypted chunks within the vector in accordance with the position of the text inside the file, when requesting a data from the file to retrieve the names of the purchasers, the data has been retrieved directly from file as text, because it is not part of the encrypted data block. On the other hand, when requested to retrieve data about the name of the issuer, the data was recalled and retrieved from the vector block.

2.7.6 Summary

XQEnc utilised both skeleton compression techniques along with vectorization method to encrypt XML data. The method is functional to assist XML query processing data in an environment of outsourcing. Furthermore, it offers enhanced safety as well as efficiency of data query processing (Buneman, 2007). After applying XQEnc, the resulting document obeys with the standard of W3C encryption, which permits diverse treatments and handlings to be utilised on various sections of the XML file. The methods can be employed among the conventional compaction methods to minimise the exchange of data overhead in the communication (Arnold, 2010).

Skeleton and vectorization showed how to protect the XML information while outsourcing can be accomplished employing XQEnc; this included the data relational technique

for storing and retrieving from outsourced services. The given method ensures vigorous security for both textual and structural information.

Message learnt and some concerns:

- There is more than one step for processing the query results, and interpreting it on both the server side and the client side. On the other hand, saving structures and other data in the client metadata makes this client site the only node that can access the outsourced server.

- On the server side, when observing the query requests and the results, it can be derived from the data structure of the XML file, and then derives useful statistical data.

- The fact that the client performs multi steps for processing the query, means that the overhead on the connection becomes higher if the retrieved data is big, and this leads to a communication efficiency drawback.

- When requesting specific information from the server, the retrieved chunk or block of data may contain unnecessary information; this will put unnecessary overheads on the network, especially if there are a large number of requests happening on the outsourced server simultaneously.

- Memory usage of the server is not monitored by this method, therefore, if the number of requests is high and the queries retrieve a large amount of data, this might overflow the memory allocations.

- Converting the original XML file into a three step transformation will add a security level by concealing sensitive data and hiding the original XML data structure; on the other hand, it might slow down the processing time, both in bidirectional ways.

- The method proposed is built on top of HTTP protocol, XPath and other query processing techniques, which are also built on top of TCP/IP protocol. This technique works only in a case of outsourcing XML database storage in a third party service. On the other hand, the concept of vectorization can be expanded to be applied on the transport layer level.

2.8 SOAP

SOAP permits both COM and objects in Java objects to communicate to one another in a decentralised Web-based and distributed environment (Dan, 2011). Furthermore, it permits any type of objects, in any language and on any podium, to communicate with each other. Currently, it has been applied in more than around 20 platforms and over 60 different languages. Unexpectedly objects all over the place, small and large, local and far-flung are competent to interoperate. Two so dissimilar types of object are ultimately capable to talk. In this section about the technology, it demonstrates SOAP at first in the wider Web-based services perspective, in universal description, discovery and integration (UDDI) as a protocol to facilitate messaging services as well as registry amongst different businesses. This review is to discourse the Web-based foundation of the rising model of ‘publish-find-bind’, along with the delivery and transport techniques in the SOAP system.

2.8.1 Network Tiers

In the development of Web-based services, three network tiers are obvious: TCP/IP, HTTP/HTML and XML. Such networks lay successively on top of one another as well as remaining compatible now (Bryant, Atallah and Stytz, 2004).

First, TCP/IP protocol is the bottom network tier, which is concerned mainly with transferring information in packets through the wire. TCP/IP is considered as a reliable protocol to transmit data across various networks (private and public), and it also stresses dependability about the physical connectivity as well as data transport. Currently, still TCP/IP considers being the key the web mostly used protocol as a transport layer and relies on it a standard to high-level protocols such as HTTP protocol (Dan, 2011).

The next network tier is HTTP protocol, which comes on top of the HTML protocol. HTML is a tier for presentation and refers to it as browser-based, sharing as well as retrieval of data. This presentation layer is stressed as GUI-based direction-finding and the handling of the layouts. In several behaviours, HTML protocol is more a demonstration than go, and needs both extensile and accurate encoding control. However, the concept of hypertext documents can be shared in an environment of browser based changed radically the method human's converse text-based data to each other. The environments of 'networked desktop', loaded with possessory of operating systems and software to rely on platforms, are gradually but definitely providing the mode in the Internet to adapt the open systems computation.

2.8.2 XML's Role in Web Services

The best solution/method to make all this feasible is the PC-to-PC communication, a region wherein XML stands out. To describe the syntax of data, XML excels is definition driven (by the employ of schemas and DTDs) and permits programmatically the data to be controlled. It

indicates that all the presumed works can be taken beyond the communication of business to business. Settlements of tags can be agreed upon, the description of interfaces can be done and standardisation can be worked out. The components of Web services are recyclable to be utilised XML standard, extensile frame of communication to be provided to C2C type of business (Buneman, 2000).

Interfaces provided by Web services facilitate the transfer of data components' data on top of HTTP along with its business logic. Web browsers can access a large quantity of information that sits under the scripts of the service provider server and in their depositaries. Web services guarantee to revive business published information, and also lay inactive in several scheme spheres (Dan, 2011).

The contributing role of XML is very important to integrate Web-tenant information into the applications of the organisation to be able to systematise the components and reform the business logic collectively. Particular services and tasks in business (comprising business and workflow logic, transaction logic, component sequencing logic, etc) can be closed in the documents of XML and integrated into business atmospheres. It permits organisations to control current resources and works and reveal the data as Web services, providing business transactions and interaction between Web services (Chamberlin, 2000).

Due to the XML being totally text-based as well as readable for humans, it is perfect as a structure of transport between Web services in a loosely coupled manner. The outcome would be: computerised business transactions enhance output, decrease costs and enhance the entire service. Standardisation makes computerised transactions of business promising, resulting in more productivity (Gregorio, 2009).

XML standard is the base of XML-RPC, and the latter was driven to produce a higher technology which is SOAP, in other logic, indications of rising standards known as electronic business XML or eb-XML.

2.8.3 CORBA's Contribution in Web Services

BEA, Sun and IBM collaborate with the very firms they fight with. The protocols of 'standardised network transport', platform-independent languages of programming such as XML, Java or industry-specific accents, and the server with open component-based computer structural design all participate in this non-propriety free-for-all. Web services current guarantees of application regarding the broader inter-operability appear as the eventual 'glue' to develop such new technologies that are interrelated, except seamlessly, leastwise without the overload that came with past technologies such as RMI and CORBA (Dan, 2011).

On the other hand, CORBA was introduced as a second approach in Web services. But, whereas it was a binary IIOP-based communications and object-oriented structure, loaded with stubs and ORB vendor-specific, Web services are frivolous, piloted in XML, based on HTTP, and totally language/platform neutral.

2.8.4 Service Providers and Web Services

Web services framework comprises of a publish-find-bind sequence, whereas the service providers develop services or data accessible to "requesters are registered service" who utilise reliable resources by binding and situating to desired services. Web services description language (WSDL), employed for requesting applications themselves, which facilitates little-

altitude procedural data regarding the desired service, granted the applications to access the schematic information of XML for data programming, making sure the exact works are raised over the correct protocols (Thornycroft, 2010).

The mechanisms of ‘Publish, bind and find’ have their particular foil in 3 different concepts except to some extent the same protocols that constitute the network in the stack of Web services: UDDI, WSDL and SOAP.

Going deep on the CORBA correlation, IIOP was contributed by SOAP. It is the mechanism of binding between discoursing endpoints. Conversely, WSDL contributes a part of ‘interface definition language’ (IDL). Herein capabilities, Web services within WSDL were described as the ports’ operations and collection. A WSDL port is similar to an interface; WSDL operations are analogous to a technique and the port introduces the interfaces to parties involved in the process of communication transversely to various platforms.

However, WSDL goes further than only being an interface, meaning language which allows you to interpret the information and the address of protocol for the desired Web services for you to distribute (Govindaraju and Bramley, 2000). Regarding WSDL, the attractive aspect is that it explains a theoretical interface for Web services whereas at the same time permitting the user, in agonising aspect to connect a Web service to a precise mechanism of transfer, such as HTTP. While the interface is considered, functions of the WSDL are a recyclable technology of Web services. By connecting to a particular mechanism of transport, WSDL develops the abstract solid. The mechanism of transport might be altered, but the payload perseveres.

Lastly, registering into UDDI is like positioning and publishing Web services. Then the service provider reveals information for connecting to the interfaces in the registry of Web-based, giving a shared index for users as well as businesses to find each other's Web services.

2.8.5 SOAP Clients and Servers

SOAP requester is actually computer software that develops a XML file comprising the data required to invoke remote methods in a distributed system. This program need not be traditional. Additionally a SOAP client could be a server application on the Web or computer desktop software (Balmin, 2005).

From SOAP clients, requests and messages are usually delivered over the HTTP protocol. Consequently, the SOAP documents are capable of traversing any firewall, allowing the data exchange across different types of servers. A SOAP service is merely a unique service that accepts requests from SOAP clients and performs as an interpreter and distributor of SOAP data. Other Web services externally may interrelate with application servers using J2EE, which processes the requests of SOAP by a range of different users.

SOAP services make sure that the data obtained using HTTP protocol is transformed to objects that the requester can interpret. Due to the complete process of communications being developed in terms of XML, Java language objects may interact through SOAP services with some other languages such as (C++/C#). It is the SOAP server's job to ensure the end points understand the SOAP they are being served.

2.8.6 Java Technology and SOAP

In accordance with SOAP version 1.1 descriptions, SOAP is “a lightweight protocol for exchanging data in a decentralised and distributed atmosphere”.

SOAP obviously is not specified for a single programming model, nor specifies the connection methods for a particular programming language. For example, in Java, it is up to the Java developers to describe the binding of a particular language. Also Java uses JAX-RPC for interconnection technique as one of the language methods.

Having comprehensively organised the stage for SOAP and defined its very important contributing part in the Web as a service it is necessary to have a close look at the functionality of SOAP. It is totally a text-based and extensible framework for allowing the interconnection between various entities, generally, objects in different platforms can interact between each other without having the knowledge of each other. Furthermore, a loosely-coupled concept environment has made it possible to interact between different applications and to locate and connect energetically to services devoid of any pre-agreed method of interaction between them.

SOAP enforces a text-based framework, therefore it is extensible, due to the SOAP clients, protocol and the servers it can develop devoid of breaking the existing applications. Moreover, SOAP is flexible in forms of assisting mediators and architecture of multi-layered structure. It indicates the nodes of processing can stay on the passageway a call acquires between the server and client. Such mediates nodes interpret the messages’ elements specified by SOAP by the headers and these headers permit clients to address which of the nodes operates on what component of the message. This mediate header working is carried out through some agreement between the user software and the mediate processing node. It facilitates the features that must be

understood for headers that permit the user to identify whether the process is compulsory or elective. For example, if the header sets to a value = 1, then the server has either to execute the mediate interpretation identified in the header or return an error.

Moreover, SOAP describes the rules of encoding data, known as ‘Section 5 encodings’ or ‘base level encodings’, a part of SOAP specification that defines it. Then it is noticeable that the encoding section in SOAP version 1.1 occupies a big part in the 40-page specification. Devoid of getting swamped intensely in the data of XML specifications, inundate continues being debilitated through professionals at the group of XML Schema, encoding as in SOAP is an accumulation of both plain and complex values.

Simple and plain values can be very simple types, such as strings, floats, or integers, or integrals as described in XML Schema description Part 2. They also comprise arrays of enumerations as well as bytes (Gont, 2008).

Complex values comprise arrays, structures and compound types as described, again, through the group of XML Schema. Finally, rules of serialisation the objects in SOAP that is identified by data encodings; it is the mechanism for marshalling and unmarshaling the stream of data on the net. It is also very significant to observe that such ‘Section 5 encodings’ are not compulsory at all, thus both the servers and clients are able to employ dissimilar standards for data encoding as far as the format is agreed on.

Lastly, SOAP develops the rules that allow both the servers and clients to perform distant process invocation employing the framework of communications. On the other hand, an oriented protocol such as SOAP-RPC operates very well using a base message of SOAP, and this is possible because of the object serialisation.

2.8.7 SOAP-RPC

SOAP-RPC is basically ‘in single direction’ message transmission from a transmitter to a recipient; however the SOAP message is frequently blended to execute the mechanisms of request/response (Wilkinson, 2005). To do RPC employing SOAP, some conventions must be pursued. Firstly, the messages of response and request must be programmed as encoded as structures. There must be an element for every input parameter of a process (structure should contain the input as part of it) and the parameter also has an identical name. In the same sense, each element must have an identical parameter for the output. To demonstrate the response and request messages, the following example demonstrates our example above:

The Request Message side:

< SOAP-ENV: Body >
 < m: Get Last Trade Buy Price xmlns: m = "some - URI" >
 < Trade Symbol > MSFT < /Trade Symbol >
 < /m: Get Last Trade Buy Price >
< /SOAP - ENV: Body >

The Response Message side:

< SOAP-ENV: Body >
 < m: Get Last Trade Buy Price Response xmlns: m = "some - URI" >
 < Buy Price > 23.35 < /Buy price >
 < /m: Get Last Trade Buy Price Response >
< /SOAP - ENV: Body >

The request calls for the method of ‘Get Last Trade Price’. Notice that the response describes an operation of ‘Get Last Trade Price Response’ (Wilkinson, 2005). A standard normal to SOAP invokes adding Response to the Request operation’s end to develop a ‘Response structure’. This structure of output comprises an element known as price, which comes back to the outcomes of the method invocation, most probably like a float.

It is also very significant that the SOAP envelope contains data types that are clearly defined, thus we still cannot recognise the type of data by looking to it. ‘Data types’ are defined by the ‘Client applications’ either generally by the ‘Section 5 encodings’, or agreed through concurred-upon agreement with service providers. In either situation, such classifications are not clearly comprised in the SOAP envelope.

Lastly, in order to apply RPC, HTTP protocol at the lower-level is required and deploying the latter as a transport protocol, the newer versions of SOAP (containing both attachments along with message) allow the employ of other transport protocols such as SMTP, FTP etc.

2.8.8 A SOAP Use Case

By observing the SOAP envelope thoroughly, it helps to stand back a little and analyse the process of the round trip that happens in the environment of web distribution (Wilkinson, 2005) for a wide review scheme that creates the abstract stamina for SOAP and its services.

- A user software someplace on the Web uses the services.
- Services using SOAP reveal the objects and its methods.
- Remote methods of objects can retrieve data from any place in the Internet.

Implementing the transitive sense to such propositions, we can realise the entire aim of SOAP and similar services: users someplace use data anyplace on the Internet.

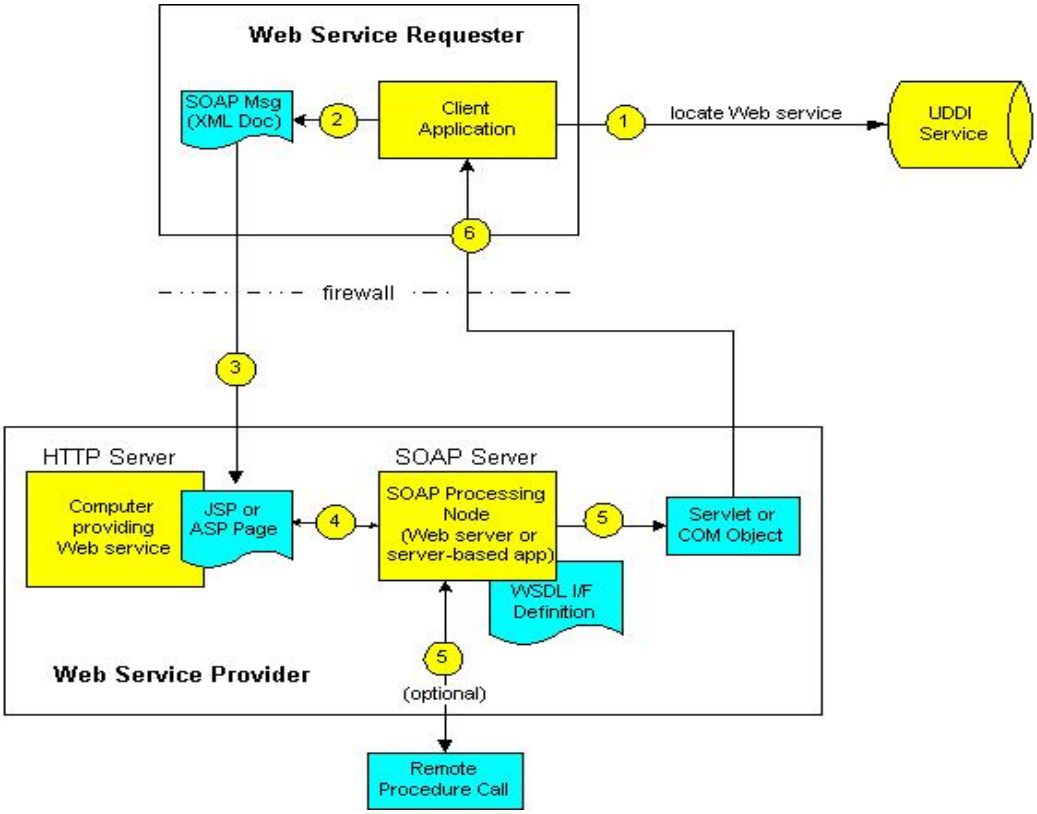

Figure 3: SOAP use case

As a thorough description for use case (Figure 3), the figure contains process numbers which are described in the process sequence as follows:

1. Services (in our case SOAP) registered in UDDI can be located and found by users. Instead of using WSDL straight away, SOAP services will mainly be installed to employ a specific type of port as well as binding style, and it will energetically arrange the service address to be requested to be equivalent to the ones exposed by the UDDI.

2. A SOAP message developed by the client application, which is a document of XML and able to execute the coveted operation of request/response.

3. Users send their message requests to SOAP via ASP or JSP or any other language to the server who is listening to these message requests.

4. SOAP service analyses the message, passes the parameters that are contained in the message and requests the suitable object's method in its area. As mentioned above, the headers' values are set to optionally process unique works before receiving the rest of the message.

5. The requested object carries out the pointed work and brings back the data or information to the SOAP service, then the latter wraps the data outcomes and forms it as a SOAP envelope. After that, the envelope is enclosed in an object of Servlet or COM or any other object form to be sent back to the original requester.

6. The requester (user) shreds the SOAP envelope object and dissolves the desired data back to the user software; in this way, finalising the request and response sequence.

2.8.9 SOAP Advantages & Disadvantages

SOAP is a layer of communication for message interaction protocol; it uses HTTP protocol as a transport layer. Also it can be applied in amalgamation with different bottom layer protocols such as JMS, SMTP, and FTP etc. Even though the most broad protocol that SOAP uses as a transport layer is HTTP, vendors start to use other transport layers such as SMTP. In general cases, messages in SOAP can pass through various transport layers before it reaches the final target.

SOAP Advantages

- Platform, language and vendor independent.

- Decoupling the runtime from the communications and the encoding environment. Platforms have no relation with the service or in the structure of the message or payload from a remote service.
- Programming language independence gave the capability to various developers (C++, Perl, J2EE etc) to contribute in a SOAP exchange, with a comparatively small barrier to development.
- Message structure uses XML to receive and send data.
- Uses an easy technique to get remote components or objects. Because it uses an XML structure, understanding the structure of SOAP is relatively swift.
- Standard HTTP used as a transport protocol layer.
- Because of using HTTP, this removes firewall complications.
- Easy to understand and deploy in comparison with DCOM, CORBA, and RMI. Since it does not transact with particular additional but essential features of remote object systems.
- Used for interchanging as protocol the knowledge in a distributed and decentralised structure.
- It can utilise different transport protocols for exchanging messages.

SOAP Disadvantages

- No protection or security mechanism mentioned in SOAP specifications.
- Message body in version 1.1 specification does not define a default encoding. An encoding is defined to be compliant in a way that any custom encoding is defined in the attribute of the encoding style or of entity aspects within the message. Therefore, in comparison with CORBA, SOAP deals with plain text to serialise the objects and not with “stringified

remote object references (interoperable object references, IORs, as described in CORBA), therefore, distributed garbage collection has no meaning".

- Remote objects do not keep condition indications at the user side (client).

2.8.10 Summary

XML is the foundation structure of SOAP for delivering different messages and creating RPCs in an environment of distributed. Text data in SOAP can be serialised regardless of a particular bottom layer transport protocol, even though the HTTP protocol is in general the choice protocol. SOAP is excellent for developing the language-neutral systems and platform that interoperate. In general, Web services and SOAP comprise the whole thing required to construct a decentralised distributed structure of application. It reduces the trouble of different platforms used inconsistencies in retrieving information through settling the dispute between object models (either Java or COM elements) or any other object modelling.

As a conclusion, SOAP is considered to be ideal as middleware between object elements of different kinds, to employ as a medium for communications.

2.9 XML KEY MANAGEMENT SPECIFICATION

Encryption as well as the signature of XML as specifications offers the mechanisms to encrypt and sign the documents of XML in the critical scenario of electronic-services and they engage the employ of crypto-keys. The necessity to incorporate digital certificates and PKIs with applications that use XML structure grows. W3C in particular is formulating an unwrapped description called 'XML key management specification' denoted by XKMS. It describes a mechanism to register as well as distributing the public keys, employed among XML encryption

and XML signature (Carlos, Miguel and Jaime, 2006). The major task of it is to permit the progress of the trusted services, which an XML-based structure is running on the keys of PKI-based cryptograph. XKMS is intended to minimise the difficulty of PKI mechanism by involving a trusted third party to rely on when interacting with various applications that adopt XML as an interchanging medium file type for every action associated with the tasks of PKI.

XKMS defines the techniques for distributing (register and process) the public keys which are completely integrated with XML encryption and signature (Nicholas and David, 2006). At an elevated degree, the protocol describes the services of predefined, communication protocols connections, a set of message formats, error models, rules for processing, and responsibilities. It is made of two main components that are explained below.

2.9.1 XML Key Information Service Specification

Key information service specification is denoted as XKISS and specifies a protocol that controls the public key information facilitating with two major services as ‘locates and validates’, employed to process as well as validate the public keys, in that order. Specifically, it also offers assistance to process ‘ds’: Key Info entity employed by XML encryption and signature. Depending on this service, applications are obviously not engaged in every action involving a communication with the infrastructure of public key, which could need some know-how regarding particular standards like ‘Simple PKI’, ‘X.509’, etc. X-KISS also permits the description of the data which provides to the authenticator ideas to employ the public key (FIXML, 2010). It is also described as a service that contains three layers; in layer-0 ‘ds’: Key Info is processed by applications; in layer-1 ‘ds’: Key Info is deputed to a service; in layer-2

‘ds’: Key Info is deputed to a service that offers further details on the data described in the ‘ds’: Key Info entity.

2.9.2 XML Key Registration Service Specification

This is denoted as XKRSS which acknowledges the PKI registration and is also in charge for the whole management of the key lifecycle. Particularly, it assists the next four major functions engaged in the process and offered through the registration process. The procedure of registration permits all the entities to register a specific key (in our case Public), attaching required data to it. Creating PUK might be carried out by the registration server as well as by a client. The service of registration may need the user to give more information to validate the request, and also checks if the user has already created the Pair-key (Private & Public) himself. The registration process obliges the user to facilitate an evidence of ownership of the matching key (Private part) (Nicholas and David, 2006). The operation of revocation permits all the entities to cancel a formerly issued key registration. The operation of recovery permits all the entities to retrieve the PK correlated with a registered PUK. Knowing that in this process may take time, along with process of Registration Service frequently to execute XML Security and invalidate the process after a request of recovering. The regeneration process permits a formerly registered pair of keys to be reproduced. The process of registration requires certifying the legality of every request of its integrity as well as legitimacy, and also requires handling verification of holding of PUKs (Yu and Liu, 2006). For that reason, the service of registration sets a policy of authentication defining a mechanism of authentication that develops offline a covert with the user.

2.10 XML-BASED ACCESS CONTROL LANGUAGE

In the initial stage, XACL were functioned to secure the resources that were themselves XML files. New suggestions however employ the XML to specify languages for showing the requirements of protection or security for any type of resources or data. The two pertinent to Web Service Policy (WS-Policy) and the XACLs are the extensile approach mark-up language (XACML), whereas WS-Policy offers the syntax to express the policy of Web services. XML-based access control mark-up language (XACML) is the effect of an OASIS organisation; they suggested a language based on XML to demonstrate and exchange and set a policy for access control mechanisms. XACML is intended to state a policy set of authorisation structured in XML alongside objects that can themselves be addressed in XML. Whereas the Web security policy and XACML split a few general features, XACML has the benefit of relishing a fundamental model of policy as a foundation, resulting in a fresh and univocal language's meaning. In the rest of the section, we describe the major characteristics of WS-Policy and XACML (Nicholas and David, 2006).

2.10.1 Access Control Mark-up Language

The main tasks provided by access control mark-up language (XACML) can be summarised below:

- Mixture of policy: it facilitates a way for aggregating ruling separately defined. Various elements can then specify these rulings on the similar resource. If a request is received to access the resource, the methods have to consider the mixture of various policies combined.

- Aggregating algorithms: given that it assists the specification of policies separately defined, there is a requirement for a technique for integrating these different rules when their

assessment is conflicting. It also assists the various aggregating algorithms, all of them representing an approach of combining manifold verdicts into a solitary resolution.

- Attribute-based: limitations or boundaries. It also assists the specification of rules reliant on attributes or properties connected with resources and subjects except the identities. It also permits the classification of strong ruling relied on common properties connected with themes such as addresses and names.

2.10.2 Setting Rules & Policies

The representations of the format used in XACML consented to control the security access by setting rules and policies as well as its function. This stage of modelling is necessary to ensure an apparent and univocal language, or else it will be subject to various assessments and interpretations. The primary goal on setting the structure of this language is to come up with clear rules, policies and set policies. Policies in XACML have, as a core element, either Policy Set or Policy. 'Policy Set' is an accumulation of Policies or Policy Set entities. Also it contains a target, an optional set of obligations, a set of rules, and a mixture of ruling algorithms. As for the target, it essentially includes a basic set of situations, actions, resources or the subject that should be met for a strategy that is relevant to a provided petition. In cases where every situation of a Target has been met, then the related Policy (or set of Policies) attempts to the petition (Al-Wasil, 2006).

The set of elements of a rule forms an object, a condition and effect. This object (target) identifies a list of actions, themes and resources, in order for this list of elements to be applied to (Nicholas and David, 2006). The consequence of the rule may be 'allow' or 'reject'. In the theme element, which can be denoted also as condition, the representation of it is a form of Boolean

expression, which by its function can treat pertinence of the policy (Biddle, Peinado and Willman, 2008). Keep in mind that the object element is not an obligatory one; policies with no specified object or a target will affect every probable request. A compulsion rule will be executed in concurrence with the implementation of an approval selection. For example, a compulsion may affirm that every approach to mechanical information must be registered. Keep in mind that just strategy that is assessed and responded of allowing or refusal may return compulsory action. This derives that the rule state is Not Applicable then the compulsion element is not relevant. The last assessment value, called the approval assessment, appended to the context of XACML information via PDP is the output or the rule as described through the mixture of the algorithm (Nicholas and David, 2006). XACML describes various mixture algorithms.

A significant option of XACML is rules that focus on the description of characteristics representing particular attributes of an action, theme, atmosphere or sources. For example, a mechanic at a factory possibly has the capability to be an investigator, an expert in a few fields, or several further job roles (Rizzolo and Mendelzon, 2001). In accordance to these assigns, the mechanic may be capable of executing various functions inside the factory. These components employ the Attribute Value element to describe the requested outcome of a special assign. On the other hand, the Attribute Selector element may be employed to identify where to put back a specific attribute. Notice that every attribute-designator and Attribute Selector elements may get back more than one value. For this cause, XACML offers an assign kind named bag, an ungraded compilation that may include repeat values for a special attribute (Rittinghouse, 2009).

2.10.3 Request and Response

Request and Response is defined as a common structure within XACML. When submitting a request by evaluation point, the interpreter interprets the request and transforms the request into a canonical structure; the transformed request will be directed to the decision point module for assessment. In its structure, it contains the three elements (action, subject and resource) and a non-compulsory element (environment). Each message contains only a single set of attributes for each element. There possibly several collections of subject assigns which is named through a class URI. A resulting component includes one or more outcomes representing an assessment. Every outcome includes three components (obligations, status and decision). The latter component identifies the status of permission returned (that is, allow, refuse, undefined, not applicable), as for the status component, if any error is returned during the assessment step, and the last component (obligations) must be filled by the evaluation point module (Rittinghouse, 2009).

CHAPTER 3

3. RIDX Design & Architecture

3.1 OVERVIEW OF RIDX ARCHITECTURE

Internet cloud communication is admittance over an unrestricted access available to the public. A secured and reliable communication environment is a big challenge for almost all businesses operating under Internet cloud. Nevertheless, Internet cloud provides a cost-effective platform for different businesses, services and type of communication to choose.

A Real-Time Interactive Data Exchange system (RIDX) virtually provides an environment to carry out secured communication between different services for transferring XML data. In chapter 2, the author has discussed several areas of cloud computing including types of services which operate under Internet cloud. Some services work as SaaS (Software as a Service) or as an integrated cloud solution etc. RIDX can simply be a combination of Software as a service and integrated cloud solution that work as a middleware for distributing XML files between different nodes, services or any domain group members.

The main functionality that can describe the RIDX system is as follows:

- RIDX is a software system that contains a pack of modules written in Java and run in any Java Virtual Machine environment. It can be run on various different platforms and operating systems that support Java virtual machine. Also it is considered as lightweight

software that can reach between 15 to 20 KLOC depending on the number of modules used in the pack of modules.

- Modules in RIDX that form the system have a flexible configuration and scalability that can be set by XML configuration file which contains the definition of pack of modules used, the order of them, and the parameters needed for each module as initial setup values. These values can be customised and vary depending on the desired setup, for example, the encryption key size can be set to 256 bit key or 512 bit key, or maybe the Ping module has a periodic packet check which is set to every two seconds or every five seconds, or a parameter to set the XML shredding message size to 1K or 5K etc. On the other hand, the order of modules can be flexible depending on the functional priority needed in the system, or flexibility to add a new module to solve a specific case requirement.

- RIDX can be installed on a node or group of nodes spread across the Internet cloud WAN and LAN that will form a domain, each node has the ability to initialise, join and leave the domain group at any time, furthermore, it can establish a secured communication and exchange data among each other. A virtual environment can be set up ahead prior to any data interchange; this gives RIDX the option to establish a multipath secured communication ground for any XML data transmission between domain members and group of nodes. RIDX architecture can be used to be a bottom most transport layer for most common used protocols, such as HTTP, or can be used for Web services such as SOAP etc.

- A failure detection mechanism that detects any unreachable members and eliminates them from the domain, also joining or leaving the domain is automatically detected; it notifies all other members in the domain.

- A shredder/assembly module which contains an algorithm to vertically shred and fragment XML files to a desired message size. Each output part resulting from the vertical partitioning process of the XML file can be formed in separate chunks, each chunk contains a group of unrelated characters that hold no useful meaning in case sniffed by an intruder of neither readable nor understandable data. This method can add another level of security in case a man-in-the-middle is trying to tap messages. The assembler module algorithm is the reverse process of shredder, the assembler rejoins the separate chunks (messages) at the receiver side into its original XML file state.

- Authentication and encryption modules are used in RIDX architecture. Authentication is used to authenticate any domain joiner and the encryption module is used to encrypt all messages between domain members.

- A reliable transmission between all members with a guaranteed message delivery and no data loss during transmission, means that the receiver will receive all messages with zero tolerance of message loss on the contrary of standard UDP. In addition, using RIDX UDP transport protocol is a multicast data transfer method, which means that every node can simultaneously send messages to multi nodes and receive from multi nodes, meanwhile, it can route messages simultaneously to other nodes.

- A module is used to tunnel and create a virtual private network between different domain members and add an additional level of security to protect data during transmission.

RIDX is an instance that resides on a node or a host at Internet cloud, WAN or LAN. It is possible to run more than one instance on the same host, each instance can be part of a domain group, this domain group usually has a mutual agreement between members (Grötschel, Alevras and Wessäly, 1998). Domains can be identified by giving a name to the PIPELINE; a pipeline is created by a process which is initiated by a domain creator (first member of this domain), by default; the domain creator will act as domain coordinator, if the domain coordinator leaves the domain, the second oldest domain member will be elected to be a coordinator, and so on. In order to join a domain, members can connect to a domain by creating a process with the same domain name. The handler between domain members is the pipeline; this means that all domain members that hold the same pipeline name can send and receive messages among each other. A domain member can be a part of different domain groups at the same time.

Pipelines are the handler between domain members, applications and pack of modules, therefore applications can connect to a pipeline to send XML data files, then the pipeline passes the data to the pack of modules to be processed until RIDX UDP transport protocol puts the processed data on to the network.

In reverse order, when the destination receives the data, the RIDX UDP transport protocol listens to incoming data via the network and passes it to the pack of modules to be processed back into the original state, and then the data are pushed to the receiver pipeline. The pipeline queues the data until the application guzzles all the data.

Whenever there is a connection request by the application to use a pipeline, the pipeline starts the pack of modules to be ready for use, and the modules will be stopped when the application disconnects from the pipeline.

The directional order of the pack of modules is defined in an XML file; this file contains the parameter list for each module, and the order of them.

3.2 RIDX VIRTUALIZATION & ARCHITECTURE

3.2.1 RIDX & Virtualization of Cloud Computing

A key element in RIDX architecture of using cloud computing is virtualization. Virtualization is “the abstraction of logical resources away from their underlying physical resources in order to improve agility and flexibility, reduce costs and thus enhance business value” (Antonopoulos and Gillam, 2010). Computing environments that are virtualized can be adapted to the required use. It can be expanded, contracted, moved, and created dynamically as required by the particular application (Wilkinson, 2005). Virtualization is a perfect fit for a cloud computing environment, as the resources need to be fluid, agile, and responsive and adaptive to the frequently changing needs of a customer. When virtualization is used, and one portion of the system is underused, it can be allocated to another operation. This improves flexibility and efficiency of architecture (Ioannidis, Keromytis and Smith, 2000).

Virtualization comes in many different forms and its precise implementation and definition can vary from user to user (or vendor to vendor). RIDX virtualization is generally considered to map a single sender into multi-connection paths or to several path alternatives into a single receiver. Virtualization in RIDX is not restricted to a centralised point of connection; it is a decentralised communication into multi-agent networking.

A virtualized network environment can allow an outside node (a customer or a service) to join and behave as if they are part of the domain group, unacquainted that they may be sharing it.

The systems are kept separate from one another. Virtualization constitutes a complete layer of a cloud computing architecture.

Administration and management of a virtualised system is a complex chore if manually processed and is paramount to the success of the system. IBM stress that automation is the key to managing a virtual environment. Automation simplifies management of the hardware layer that make available the needed resources. The two most recurrent tasks performed in a cloud computing data-centre are on boarding and off boarding of applications (dynamic provisioning). On boarding is the installing of RIDX and configuring it and additional modules added to the pack of modules in order to be prepared obtainable to perform helpful work. Off boarding presents to the stages important to recoup automatically to obtain different aims. When performed manually, these tasks can be time-consuming. Automation makes these tasks efficient. Another key area that is automated is reservation and scheduling of resources (Ioannidis, Keromytis and Smith, 2000).

Along with automation, a self-service API access is a vital part of the management architecture. This handles requests and automates management. It allows customer/service to access through a Web service or through direct API listeners to request a service. Monitoring is an essential part of the system management architecture as it collects real-time and historic data and measures performance. Capacity planning is an important element that needs to be managed. This becomes complex in a virtualized environment, because it is less predictable in terms of size of domain members, and the size of XML data files usage than a static environment.

3.2.2 Topology

The Real-Time Interactive Data Exchange (RIDX) system resides on a host within the domain area (group of nodes that share same domain/interest) in the Internet cloud (Ho, Vincent and Cheng, 2009). Each RIDX system contains modules, processes and listeners, some responsible for keeping track of domain members, some for delivering and receiving messages, others for routing messages and for shredding/joining messages etc.

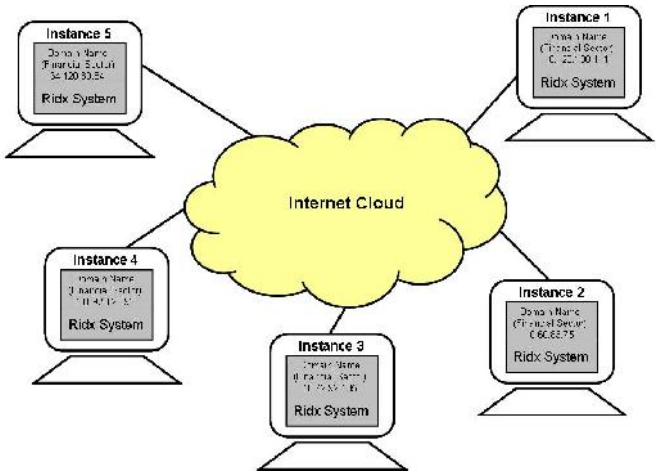

Figure 4: RIDX instances distributed in the cloud, domain interest is 'Financial Sector'

3.2.3 Architecture

RIDX architecture is divided into three major parts:

1- Pack of modules containing classes to process XML data which includes data encryption, XML vertical partitioning, data assembling, message sequencer, data flow control and group of modules that administer domain states and membership control.

2- Transport protocol which consists of three layers, the bottom most layer is tunnelling; on top of it a router module, and then the higher transport protocol layer is RIDX UDP protocol.

3- The interface (API) which provides the accessibility to different types of applications to enable users to access RIDX pack of modules in order to facilitate the pack of modules to interact and work as a middleware software for exchanging XML data between different parties.

Figure 5 shows the hierarchical layers in the RIDX system, for example, Instance (1) shows that the highest layer is the application layer; applications can be any software application that uses an XML file structure to interact with other applications. The application layer can also be any Web service on the Internet, such as SOAP, CORBA or various application servers etc. These applications can integrate and communicate with RIDX through RIDX APIs. The next sections explain about different layers as follows:

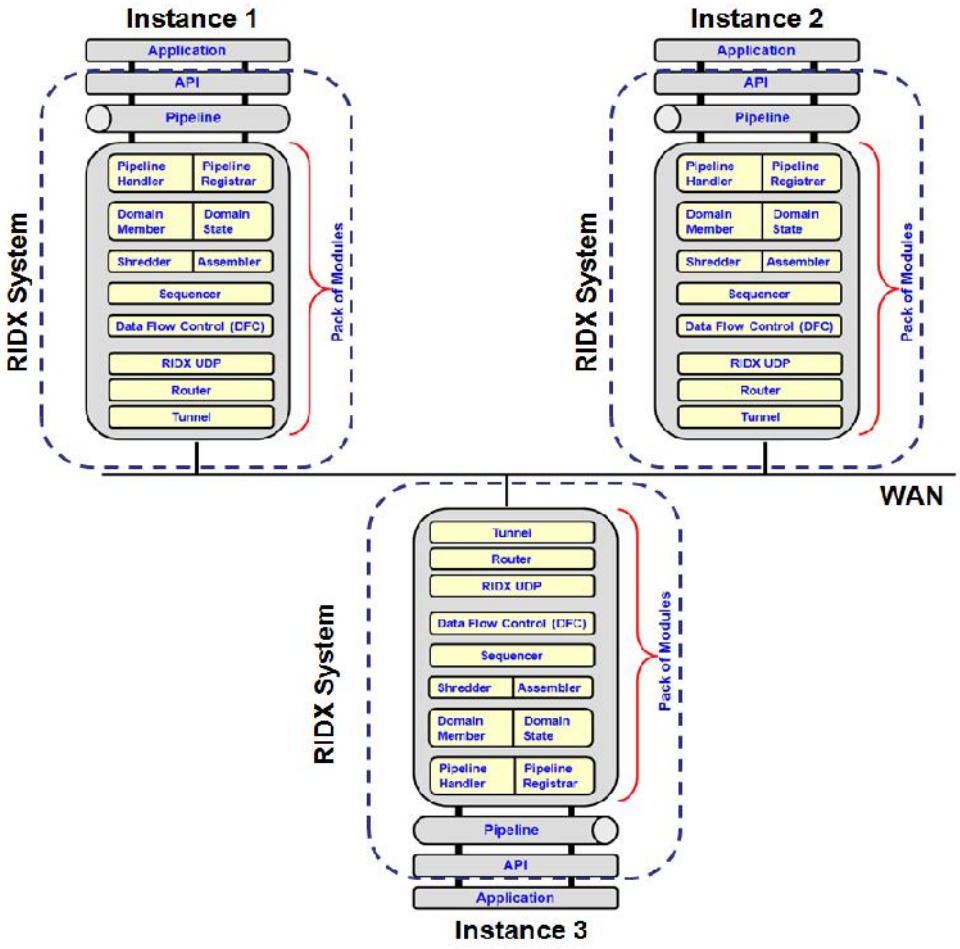

Figure 5: Instance structure & different modules which are used to facilitate major functionality

3.3 THE CONCEPTUAL MODEL DESIGN & COMPONENTS

A pack of modules facilitates the functionality of the system; the Application Programming Interface (API) is the interface between the RIDX system and the application software layer accessing it to allow using various services and resources, along with different modules and protocols. Each XML file sent by the application/user through a pipeline/API will be processed by the hierarchical layered modules until it reaches the lowest layer (UDP or tunnelling layer). Messages received by transport layer during transmission session will process and hand over each message to the next upper layer through the pack of modules up until it reaches back to

the application layer. System parameters, configuration setup and initial values are saved in XML file structure; the values that are kept in this file can drive the system to perform the desired outcomes from the system, for example, the message size can be controlled by setting (Partition Size) the variable to a desired shredding size, or the number of credits in the Data Flow Control module can be set to a certain number in order to control the data flow etc.

The following modules describe the components from top layer to bottom used by the system as in Figure 5; the transport layer, which consists of three modules (RIDX UDP, Router, Tunnel), will be discussed in chapter 4:

3.3.1 Listeners/interfaces (APIs):

- **Message Handler:** whenever this listener receives a message, a push style event handler notifies the receiving of the message, number of methods is invoked to determine the state of the message in order to be processed. When a message is received, the method receive() will be called. Both getState() and setState() methods are used to get hold of domain member's state. In addition, when a message is received, a byte buffer is allocated for this message, and then appends this buffer into its queue, this queue is dedicated for input messages (received messages), and another queue is dedicated for output messages (sent messages). Append process eventually is a copy of the first initialised byte buffer, this copy is done in sequence to un-serialise the message in the copied buffer at the input queue instead. Ultimately this is prepared in order to free the message handler thread to receive new messages. This is due to the time needed to finish the serialisation/un-serialisation process, but doing so gives the ability to continuously pass the un-serialised messages to the modules above to be processed.

- **Domain Members:** each domain member has a list of active members that is joining the domain, whenever a new member asks to join or leave the domain; a snapshot is taken of the list and distributes it after registering the new status to all domain members. The domain coordinator is in charge of accepting/leaving and announcing the new snapshot.

  When suspecting one of the domain members of crashing without announcing leaving the domain group by disconnecting from the Pipeline, the suspect() call-back is raised.

  The viewAccepted ( ) method will notify the requester and all the other members in the group that a new domain member has joined the domain and been accepted. At the same time, it will notify domain members that the domain member has left (disconnected) from the domain group.

  In case of two or more requests received from different new requesters to join the domain, the block() method is invoked in order to block all the request messages until the current request is determined and finalises the status of the new requester and synchronises the status with all domain members.

- **Pipeline Registrar:** in order to send and receive messages, the user needs to register in a pipeline, the pipeline name identifies the domain and the coordinator holds the state of the pipeline whether it is active, inactive or closed.

  To register in Pipeline, a method Pipeline.setPipelineListener ( ) is used for implementing a class PipelineListener ( ) in order to obtain the state of the pipeline and any other information to declare the pipeline state. A call-back will be raised whenever a pipeline is opened, disconnected or closed.

- **Pipeline Handler:** the pipeline handler passes all XML files from the application to the pack of modules or from the pack of modules to the application.

3.3.2 Pipeline

Pipeline is similar to a socket; a new joiner must create a pipeline and join a domain in order to send and receive messages, a process will handle creating a pipeline. Connecting to a pipeline will give a name of the domain that would like to join. Whenever a pipeline is active and in a connected state, it is always allied with a particular domain. A pack of modules look out so that pipelines that are holding the same domain name can find each other. A finder module is loaded when a pack of modules are loaded, the main job of it is to search and find an active member on the net, it triggers a message with the help of Ping module to locate any member on the LAN. If there is any positive response from a member during the Finder process, the member will respond to the request by sending back the status of it along with the coordinator IP/Port address. The Finder module will extract the coordinator address and pass it to the AML module for further communication to join the domain.

A pipeline has three main states (Not Connected, Connected, and Closed), sending and receiving messages is only compelling when the state of the pipeline is connected, the next figure shows the states and the process of changing the state:

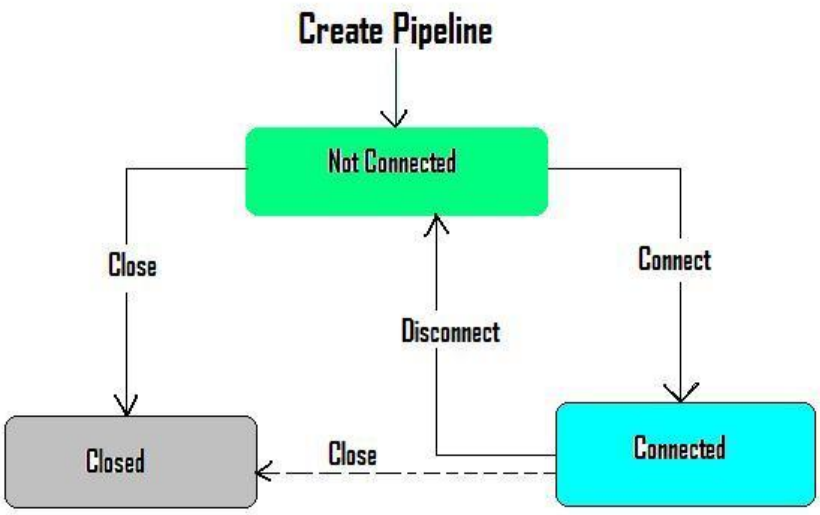

Figure 6: Different states of the pipeline & the process of changing between them

Whenever a successful connection has been made, the system will prepare the pack of modules for this connection to be ready; an XML file grasps the configuration parameters including the desired list of modules needed for the system along with the parameter list for each module.

3.3.3 Shredder & Assembler

The shredder & assembler modules are one of the defence mechanisms of this architecture. The first purpose of these modules is to break down the XML file into several chunks (small pieces) by applying a vertical partitioning, adding an identifier to each chunk, then attaching this identifier to a message along with the data; the ID can be added separately either by adding a header to the message or embedding it in the message body. At the final destination (Receiver), all messages are re-assembled again.

The size of the chunk is predetermined in the configuration file, this decision depends on what level of partition size needs to be applied. For example if a high level of security is required depending on the sensitivity of the data, a smaller size of each chunk is imposed.

Although fragmentation is part of a standard TCP protocol by its nature, it has several concerns:

- The performance and inefficiency can be affected negatively when reassembling fragmented data.
- The performance and high cost of retransmitting can be affected when number of lost fragments is high.
- In a standard UDP case, the performance can be affected when using more resources because of unfortunate choice of datagram size, this happens because of lack of guessing the Maximum Transmission Unit (MTU) size. MTU size varies depending on the hardware, router and operating system that has been used given that when UDP sends a message, it does not expect any feedback from the receiver, and therefore, the fragmentation is controlled by the operating system and not UDP protocol itself.

The Shredder algorithm reads the XML file and splits it into several 1K size chunks as a default system parameter partition size. The 1K chunk size is preferred to be used to keep the whole message that might include additional headers under the standard MTU size (1.5K), by doing this, the fragmentation of messages larger than the MTU size can be avoided and this eliminates any possible retransmission cost and enhances transmission performance (see chapter 5). Eventually, the number of partitions depends on the whole XML file size. The process starts from top down vertically to form an array that might be similar to a set of byte arrays of character type. A dataset of $X = \{X_0, X_1, X_2...X_n\}$ where X represents XML file and $X_0... X_n$ represents numbers of partitions generated from partitioning File X. In the next figure (Figure 7), which illustrates an XML file shredded into 15 chunks, an ID is assigned for each partition in order to preserve the order of messages at the receiver side.

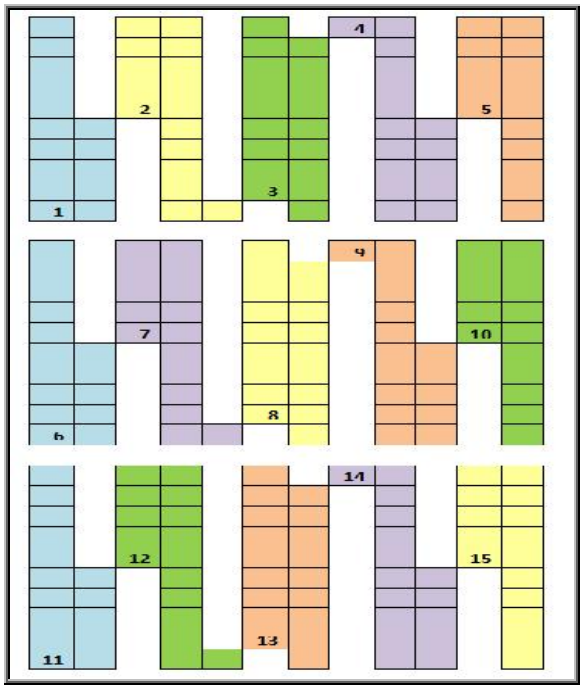

Figure 7: XML shredded file example

At the receiver side, the assembler reconstructs the file by verifying and reassembling chunks according to message sequences. The message sequence is unique in conjunction with Sender ID (obtained from Active Member List), and a process ID which is unique generated by shredder layer and assigned to a specific XML file, and a partition ID which is unique per partition sequence number within this specific XML file process. Any additional information that needs to be included can be attached to the message header. As shown in performance results in chapter 5, the best results are obtained when keeping the chunk/partition size 1K or under per message size.

3.3.4 Sequencer

In this module, a reliable and vigorous mechanism is applied. Among them included the use of sequence number to ensure the orders when receiving the packets, also to detect the

duplicate RIDX UDP segments, as well as employing the checksum technique to detect errors, and enforcing acknowledgments and timers for leaks and delays.

When the connection is established, initially the sender and the receiver exchange and swap sequence numbers between them, this identifies and counts the data segment in the byte stream which has been received through the application layer. A pair of numbers is always formed as part of the packet, one for sequence and one for the acknowledgement. At the sender side, the sequence number reflects the sequence of the data being sent, as for the acknowledgement number reflecting the receiver's sequence number. To sustain reliable communication, a recipient replies back the location of the segments received representing a part of the continuous stream of bytes.

One of the keys to preserve the forcefulness and security of RIDX communication is the choice of initial sequence number (Initial Sequence Number - ISN). The sender computes the 16-bit checksum by counting the summation of the data and header ones' complement values, and appends this number in the transmission segment. The receiver recalculates the checksum of the header and data, and simply calculates the summation of the 1s complement with the checksum included; the outcome must be zero. Therefore, it is presumed that the segment has arrived intact and without errors, and needs to mention that checksum in RIDX UDP protocol including the destination and source addresses. This gives protection adjacent to misdirected packages for errors in addresses.

The acknowledgments (ACKs or Acknowledgments) of data between the sender and the receiver can benefit from it; the sender in particular can understand network conditions between them. When a timer is employed during the transmission, both the sender and the receiver can

adjust the performance of the data flow. To accomplish high performance and circumvent any congestion in the network in RIDX, there are some techniques used by RIDX UDP protocol for controlling the flow of data (the sender sends as fast as the receiver can receive). A dedicated module for controlling data flow is produced and discussed in the next section, and there is a sequence number generated by each side, thus helping to establish that connections cannot be falsified (spoofing).

When a sender starts transmitting messages, the first message is given an initial sequence number, the receiver tags this message as a starting sequence number and creates a negative acknowledge window. On the sender side, a specific timeout is set to monitor messages which is delivered to the receiver. If the receiver did not acknowledge receiving all messages, the sender resends messages that have not been acknowledged, a specific timeout is set in case during transmission the sender has crashed or lost connection and eventually stops an endless retransmission process. The same case is monitored in the receiver side; when the sender requests retransmission of non-received messages, a timeout is set to control endless retransmission requests and considers whether the sender is crashed or lost connection, therefore the file transmission is considered as failed.

3.3.5 Data Flow Control

Data Flow Control is responsible for controlling the speed of transmitting when the receiver is slower than the sender. When messages overflow the receiver, it starts to drop any new messages, and this will cause the sender to retransmit the dropped messages, leading to a higher cost and lower performance. There are several techniques to control data flow used by standard TCP protocol such as stop-and-wait, sliding window etc. None of these techniques can

be deployed in RIDX communication structure because these techniques assume a Unicast one-to-one connection and will not work within Multicast RIDX architecture. Therefore another approach has to be implemented; this approach should cope with the characteristics of the connection topology (1-N, N-1), where one sender sends messages simultaneously to number of RIDX routers (N), then number of RIDX routers (N) routes back the messages to one receiver. The receiver receives number of N messages simultaneously; the sender processes a group of N messages until the whole XML file is transferred.

Consequently, this process can cause an overflow state on the receiver side over time. A simple technique is implemented in RIDX to overcome this problem by using a data flow control (DFC) module. For instance, one of the techniques used is 'Bank Account', which controls the transmitter to send information within the receiver's buffer and flow control algorithms, such as the algorithm for congestion avoidance, the slow start (Slow-start), the fast retransmit and the fast recovery (Fast Recovery).

This module is similar to an acclaim bank account transaction; each member holds a number of credits parameter, this parameter is defined in the configuration file (see Figure 8) and the account credit number can be preconfigured to meet certain conditions. When a sender initiates the process and starts sending messages to a receiver, it initiates the maximum number of credits defined in the system specifically for that receiver. In each new process of transmission, the credit number is initiated again along with it. When the sender starts sending messages, the sender decrements from a maximum bank account value by 1, if the account value is decreased below 0, the sender stops sending messages. On the receiver side, whenever a receiver receives a message, it reimburses the sender bank account value by 1. Therefore, the

sender keeps sending messages as long as the credit number is greater than 0. Messages in the receiver side will be processed and queued in the pipeline until all messages are received and reconstructed back to their original XML file structure.

3.3.6 Identifier

Each domain member has to be identified, when using UDP transport layer, the IP Address is the unique identifier for each node plus the port number (Socket Number) on which the message handler will be receiving messages. If it is desired to run more than one RIDX instance on the same node, then all instances have the same identifier (IP Address) except for the port number. Therefore port numbers differentiate between multiple instances on the same node.

In order to determine if the IP address (Identifier) is a multicast IP address, RIDX must have solid implementation methods to sort and compare the addresses and to allow all probable types of addresses (e.g. ATM etc), RIDX UDP protocol generates the actual implementations of addresses and the interface should not be implemented directly to preserve the unique identity of RIDX node.

3.3.7 Active Member List

When Domain Coordinator (First Member) initiates a Pipeline, an Active Member List is created, this list will be identified by an ID, and it contains all the existing active domain members. Whenever a new joiner to the domain is registered into a pipeline, the domain coordinator will release a new AML (Active Member List), then the coordinator will distribute this list to all domain members. The list identifier (ID) identifies the version of this list and sequences any new ID whenever a new list has issued. In the same sense, whenever a domain member leaves the domain, either by disconnecting or leaving or even by crashing, the

coordinator will issue a new AML and all domain members will be notified of the new AML sequence along with the new Active Member List.

The Active Member List contains current and active members which forms the domain; active member means that the member is reachable with a positive response when firing a Ping message and the member has RIDX modules including the initialisation of RIDX Pipeline along with all modules needed to communicate with other members. The active list has two main columns, one column contains the IP/Port address for each active member, the second column contains the current coordinator for each member, so each row contains a pair of one of the domain members and its coordinator.

Maintaining the AML is very important because the list is the main and primary reference to maintain the communication modules used by RIDX, for example, RIDX Router totally depends on AML to control routing messages between the sender and the receiver. The Authentication, Encryption and Tunnelling modules depend on this list always being maintained and periodically checked for continuous updates. The coordinator plays a major role in governing this process and maintaining it periodically.

3.4 MODULES & GENERAL WORK OF RIDX

3.4.1 Initiation of Process

- **Initiating a Pipeline:** initiating a pipeline can be done in two ways, one by creating it directly using a constructor (e.g. new Pipeline( )) which is an instance of a sub-class of (Pipeline), the other way is to create a Pipeline Factory upon request. The following is by using the first method to create a Pipeline: Public Dpipeline (String props) throws PipelineException{}.

The configuration parameters for this method are read from an XML file which contains all the parameters needed to configure the pipeline. If the pipeline was not created, an exception will be thrown, a wrong or missing parameter is mostly the possible reason that causes an exception. The RIDX UDP XML configuration file looks as follows:

```
<config>
    <RIDX_UDP
    mcast_addr="${ridx.udp.mcast_addr:182.0.0.10}"
    mcast_port="${ridx.udp.mcast_port:45588}"
    mcast_recv_buf_size="25000000"
    mcast_send_buf_size="640000"
    loopback="false"
    discard_incompatible_packets="true"
    max_bundle_size="64000"
    max_bundle_timeout="30"
    use_incoming_packet_handler="true"
    ip_ttl="${ridx.udp.ip_ttl:2}"
    enable_diagnostics="true"
    use_concurrent_stack="true"
    thread_pool.enabled="true"
    thread_pool.min_threads="2"
    thread_pool.max_threads="8"
    thread_pool.keep_alive_time="5000"
    thread_pool.queue_enabled="true"
    thread_pool.queue_max_size="1000"
    Aux_thread_pool.enabled="true"
    Aux_thread_pool.min_threads="1"
    Aux_thread_pool.max_threads="8"
    Aux_thread_pool.keep_alive_time="5000"
    Aux_thread_pool.queue_enabled="false"
    Aux_thread_pool.queue_max_size="100"
    Aux_thread_pool.rejection_policy="Run"/>
    <PING timeout="2000" num_initial_members="3"/>
    <FD timeout="10000" max_tries="5"/>
    <VERIFY_SUSPECT timeout="1500"  />
    <pbcast.acknak retransmission="true"
    exponential_backoff="150"
    retransmit_timeout="50,300,600,1200"
    discard_delivered_msgs="true"/>
    <pbcast.router stability_delay="1000"
     desired_avg_router="50000"
     max_bytes="1000000"/>
    <AML_SYNC avg_send_interval="60000"   />
    <pbcast.AML print_local_addr="true" join_timeout="3000"
    view_bundling="true"/>
    <DFC max_credits="500000" min_threshold="0.20"/>
    <Shred shred_size="60000"  />
</config>
```

Figure 8: RIDX system configuration file

Every element in the configuration file delineates one of the modules in the pack. For example, in Figure 8, a parameter is set to identify the node by assigning IP address and port number in ‘Multicast’ address and port parameters. The receiving and sending buffers are set to a size in bytes. Two main thread pools are set which include maximum and minimum number of threads per pool, and the policies related to it. The main thread pool is for handling RIDX data messages, the Auxiliary pool handles messages which are

marked as Aux messages, for example, the Ping heartbeat message is marked as an Aux message, therefore, it will be processed in the Auxiliary pool threads. This method is very helpful for processing RIDX data messages quickly and separately from utility messages, because data messages do not need to wait until utility messages are processed especially if it is from the same sender. Ping, Failure Detection, Active List, Authentication and Synchronise encryption keys are considered as part of utility messages, therefore it will be processed separately by Aux Pool. Failure detection and Ping parameters are set to timeout in micro seconds to determine when nodes in group are continuously available or crashed. The last two parameters in Figure 8 are set to initialise the data flow control maximum credits and the shredding size of each message when processing the shredding module on XML files. The XML configuration file is used to preserve the independency and scalability, each module is implemented by a separate Java Class; also each element can represent a layer in RIDX.

- **Pipeline Naming:** when a pipeline is created, a Universally Unique Identifier (UUID) is assigned to this pipeline, instead of using the pipeline address; we can set a more useful logical name for this pipeline. For example, the UUID creates an identifier such as: ‘29f96f2d0a77-9a0e-9581-7804-ae25e7c3’, instead, we can set a more meaningful name that might illustrate the function of the pipeline, for example: ‘No1-domain-accounting’, this name is more understandable than the UUID. To set the logical name, a method is invoked to do so. The pipeline will hold the logical name until the pipeline is destroyed. Setting the logical name happens before connecting to a pipeline.
- **Connecting to Pipeline:** to join as a member to a domain, it needs to specify the name of the domain: Connect (String DomainName) throws PipelineClosed.

A domain in RIDX is the group of members connecting to the same Pipeline name. When a user or client invokes this method, it checks the state of the domain if it already subsists. If it exists, it will check the state of the domain whether the domain is in close state or open (see Figure 6). If the domain name does not exist, then it will create a new domain name and the client will act as a coordinator for this domain member, therefore, the coordinator will be responsible for governing the initiated new group and controls the new members' joining process.

- **Retrieving Name & Address:** generating an address generally happens in the lower bottom layer (RIDX UDP) module. If the state of the pipeline is closed or disconnected, we cannot retrieve the address, otherwise we use this method to retrieve the local address and the name of the domain.
- **Sending Multicast Message:** sending a message in multicast mode needs to specify the source and destination addresses (Sender & Receiver Addresses) and also needs to serialise objects, then build a message according to the selected parts of the XML document that has been shredded from the shredder/assembler module. Then it sends the message; exceptions occur if the pipeline is disconnected or closed.
- **Message Receiver:** pipeline can accept registration by message receiver, as a replacement for pulling out a message from the pipeline; this will save one thread to serve this issue, as a result, queuing messages is not the pipeline responsibility to maintain it, because it can get large if the receiver thread cannot process the data message swift enough. To register a Receiver Interface in a pipeline, all the call-backs will be invoked in case of AML changes or receiving messages.

  The pipeline can hold different types of messages, one for XML data message, one for

updating the Active Member List, and one for exchanging the State Transfer of any RIDX node.

- **Disconnect from Pipeline:** the following method is used to disconnect from the pipeline: public void disconnect(); when invoking this method, it will inform the coordinator by sending a disconnect message, this will remove the user (pipeline address for this user) from the domain members. The coordinator will issue a new member list and distribute it to all domain members. The pipeline will be in a disconnect state after a successful disconnect, this can be reconnected if the user requested it to do so.
- **Close a Pipeline:** closing a pipeline means that all the resources for the RIDX modules will be released and destroyed; no more possible operations can be made if the state turns to close. The Java Runtime system will free all the resources and invoke a Garbage Collection process.

When requesting to transfer XML data initiated by the user at the data application layer, the system will initiate the first step by creating a pipeline process; this pipeline process is the handler between the user application layer and RIDX system. A set of properties belonging to the pipeline process is created at the time of initiation and then registering to a domain by specifying the name of this domain, e.g. Domain name (‘ABC’), every member in the domain ABC will see each other. A request from the new joiner to get AML (Active Member List) from the coordinator, the coordinator accepts the new member and provides the new AML to all domain members.

The XML file sent by the user will be shredded into several chunks; the size of each piece is specified according to the parameter set in the RIDX XML configuration file. Each piece is encrypted and encapsulated in the RIDX message structure, each RIDX message contains all the information needed to be transferred until it reaches the final destination. The system will choose a

different path for each chunk according to the Active Members List and transmit simultaneously in a UDP multicast reliable transmission. The next graph represents a sample of transmitting a XML file after shredding it and sending each partition into different paths from sender to receiver through virtual RIDX domain group, several modules will be invoked before the transmission operation starts, for example, the module Shredder/Assembler vertically partitions an XML file into chunks and prepares them into a table in the buffer ready to be sent, then the table is handed to the next layer which is the Sequencer to sequence each chunk for maintaining the ordering and acknowledging messages by the receiver. The Sequencer holds the prepared sequenced chunks into a table in the buffer to be handled by the transport protocol to form all messages that will send all messages in a reliable multicast transmission.

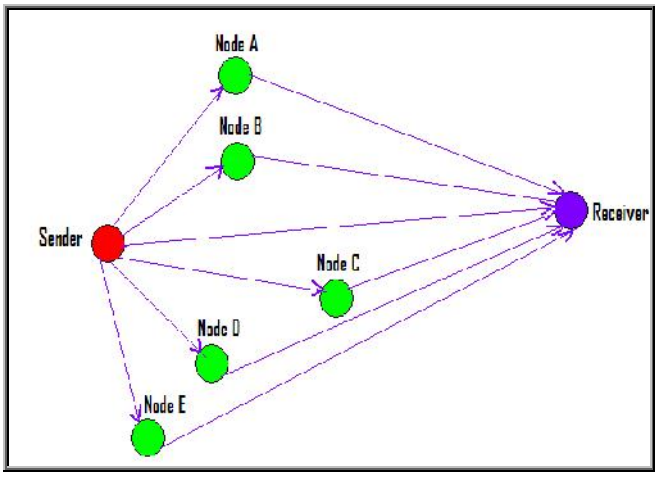

Figure 9: RIDX Internet cloud domain members: domain name 'ABC'

At the final destination (Receiver), all messages are sequenced; therefore messages will be reordered, decrypted and joined/merged together until the XML file is constructed again into its original state. In case of any packet loss, the receiver requests the sender to re-transmit the lost messages. As in the figure above, Nodes (A, B, C, D, E) work as routers to route all messages to

the final destination.

3.4.2 Data Encryption in RIDX

The main methods used to encrypt messages are Asymmetric or Symmetric encryption methods. The two methods vary from each other in the usage of the encryption key. For example in symmetric encryption, the same key is used for both decryption and encryption of the data, whereas in the encryption using asymmetric algorithm, the keys are in pairs but are not equal, one key is applied for encryption and another key for decryption. Symmetric encryption is considered faster and easier than asymmetric encryption. The security strength of the encrypted data is related to the length of the key. Asymmetric encryption has one private key that is recognised just to a specific user who is using it to encrypt information; this private key should not be distributed and shared with any other users, and one public key that is known between different users (Matuszek, 1999).

Secret and private keys are very similar and are often used interchangeably. The difference is that secret keys are used for both encryption and decryption, while a private key is part of the public/private key system and is used only for decryption (Cryptography, 2005). In both cases, the key may be known only to a single person or a limited group of people in order to keep the key secure.

In RIDX, communication between domain members can be encrypted, this can be done in two scenarios, one is to assign the encryption process to the coordinator, or it can be preconfigured within the RIDX configuration setup files. The following will illustrate the two scenarios provided:

Scenario 1: (Asymmetric RSA)

The coordinator plays a primary role in controlling the process of distributing and synchronising encryption keys between all domain members. This mode depends on using customised encryption key size and configurable algorithms, and the key size according to the encryption methods used, for example, a 512 bytes as a default key size in case of Asymmetric encryption algorithm used, and a 128 bytes size for symmetric algorithms. When the Asymmetric algorithm has been chosen to be used in RIDX, then the configuration parameter needs to be set to a default key size (512 bytes), whenever a new domain member requests to join the domain, the coordinator will issue a new Active Member List (AML). The joiner requests a private key after sending its own public key to the coordinator, the coordinator generates a new private key and encrypts it with the public key which was sent by the requester (new joiner) and then sends it back to the joiner. The new member decrypts the key and extracts the private key to be installed and used for further data and communication encryption. Whenever there is a new AML issue, this process is repeatedly triggered again. During this process, if any messages have currently been sent, it will be queued in the outgoing and incoming queues until the coordinator regenerates the new key and synchronises the public/private keys with the new joiner and all other domain members.

Scenario 2: (Symmetric Triple DES)

Java (Sun – Oracle) provides a library for cryptographic support through JDK standard. They provide two options for applying cryptographic methods using a key store file basis, one is abbreviated as ‘JKS’ (Java Key Store), and the other comes as an extension and it

is abbreviated as ‘JCEKS’, (Java Cryptography Extension Key Store) as part of Java Cryptography Extension library ‘JCE’. JCEKS uses Triple DES (Digital Encryption Standard) encryption which is considered as “stronger protection for stored private keys than JKS” (Java SunPKCS11, 2001). RIDX can be set up to use triple DES algorithm with the implementation of password protection based encryption (MD5) algorithm. Using these methods in RIDX is considered much simpler to implement than the Asymmetric technique because it eliminates the complexity and the overheads of exchanging the encryption keys between domain members, therefore, it is faster in processing time. The key store file contains the secret key, which is generated by API key tool method provided within the Java JDK. This file must be synchronised with all domain members and all members must have the same file. This encryption technique can be used in any part of RIDX modules, and can encrypt any event of type MESSAGE without the need of a coordinator.

In the two scenarios, both parts of RIDX message (Data & Headers) can be encrypted, the encryption key version and the encryption identifier and all the information needed by the receiver can be attached to the message as a header.

3.4.3 Authentication in RIDX

The purpose of Authentication mechanism in RIDX is to authenticate any request of joining the domain and reject any attempt of un-authorised joining requests. This module is processed before the Active Member module (AML), whenever a joining request is received. The Authentication module initiates the authentication process, if the answer back from this module returns ‘Allow’ as a result, then it will pass the request to the AML module for further

processing. Otherwise the authentication module will result in a reject state and discard this request and send back the security exception error along with the type of error.

The authentication process has some differences in the techniques used as in the certificate authorities approach. For example, the Public-Key Infrastructure (PKI) is a method of verifying users on a network, while a digital certificate is a reference from a neutral company that confirms the identity of an Internet site. Certificate Authority (CA) for instance (VeriSign) provides digital certificates, as for the registration authorities (RA) which operate to recognise the user and be a reference to it, and use a directory that holds the certificate and can revoke a company's digital status. The PKI technology is at the core of the digital certificates used in almost all transactions on the Internet. The PKI uses a cryptographic key pair, one of which is public and one which is private, to authenticate the owner of the certificate (Stanton, 2005).

The authentication mechanism used in RIDX is a software security token, which is simply a text encrypted string which is considered as a key for accessing the system. This key is a way to prove that the user is the one it is claiming to be. Tokens in Java are used as part of the utility library under the name 'Auth Token'. The Authentication Token utility is an abstract class used to generate and validate authentication tokens, and also using Java tokens is considered as a single step for the user to sign into the authentication system. An external file contains the authentication string token (credentials), this file has a limited access and can be accessed only by a certain RIDX process initiated by authentication class, and this process can extract the credentials from the file. The authentication token file is protected and distributed within RIDX configuration files.

Security threats can always be a concern when trying to establish a secured communication; therefore one single authentication file and unencrypted token string can always

be a concern, especially when the hacker can sniff or spoof the message or have unauthorised access to the authentication token file. Hence another process is added to increase the level of security to the authentication process of member joining request. Encryption methods can be used to encrypt the token string while sending it for verification along with the certificate implementation. This identity certificate combines and binds the identity of a member along with the public key to form a digital signature. For example, certificates contain the domain member information which forms the identity of it along with the encryption/decryption public key. The next algorithm illustrates the authentication process using digital certificates and authentication tokens:

- A new member wants to join the domain by sending a joining request message to the coordinator along with its digital certificate.
- The domain coordinator checks the chain of the certificate and verifies if it is a valid certificate and decides if the member is trusted to continue with the joining request or not. If the certificate verification results return a negative state (Not Verified), then the coordinator responds back to the requester with a negative response with an exception error and no further action taken.
- If the coordinator verifies the certificate, then it will send an authentication request to the new joiner, which also contains the coordinator's certificate to be verified by the new domain joiner, the message encapsulates the private key that will be used by the new joiner to encrypt all the next messages. The new joiner decrypts the message using the public key and verifies the certificate and extracts the private key.
- The new joiner responds to the coordinator and sends the authentication token, and encrypts the message using the private key which was sent by the coordinator.

- The coordinator verifies the authentication token string, if it is successfully authenticated or not, and then the coordinator passes the request to the Active Member List module to process the joining request.
- The AML module prepares a new list of all members including the new joiner; the list contains two columns paired with the coordinator's IP/Port address and all other member IP/Port addresses. The AML distributes the new public key encryption and synchronises the public key between all domain members including the new member.

The authentication process including the Public/Private encryption has five to six steps to authenticate, but it is acceptable, first because it preserves a higher security control and second this process does not happen very often, and triggers only when a new member wants to join the domain, and third, whenever a member joins or leaves the domain, RIDX ensures to issue new encryption keys to all members, in this case the new joiner cannot decrypt old messages and the member that has left the domain also cannot decrypt any current messages.

CHAPTER 4

4. RIDX Transport Layer

The RIDX Transport Layer resides on the bottom of the RIDX pack modules (see Figure 5). It is responsible for sending and receiving data through LAN and WAN networks. This layer consists of three main modules and a set of supporting modules to implement and support the fully functional transport layer. The main modules are consequently ordered from bottom to top as Tunnelling, Router and RIDX UDP. The tunnelling module level is the bottom-most level; it encapsulates all messages coming from upper modules by multiplexing/de-multiplexing when sending data from one end to another. More discussion is made about Tunnelling, Routing and RIDX UDP in this chapter along with the supporting modules and their functionality.

In chapter 2, the author has discussed some Internet cloud services (SOAP, CORBA etc) and Access Security Modules (XACML etc), they all use the XML file structure as the intermediate, and they adopt XML as a communication file structure between these systems and services, also some of these systems and services work on top of TCP/IP Protocol or Standard UDP protocol directly or indirectly. For example in case of SOAP services, SOAP uses HTTP protocol as a communication method, whereas the latter uses TCP/IP protocol as a bottom-most transport layer for communication. In some other systems such as multimedia streaming services use standard UDP protocol as the bottom-most transport layer for its communication. One of the techniques used in Media Resource Control Protocol implemented a different method by using Real-Time Transport protocol (RTP). RTP protocol is considered as a sub-protocol of UDP (Internet Engineering Task Force (IETF) RFC 3550, 2003), there are two concerns of using this

protocol first as it “can tolerate packet losses” and second, RTP protocol is formed within the structure of standard UDP protocol, this made RTP protocol inherit the characteristics of UDP protocol.

The motivation behind RIDX is to create a virtual network structure throughout Internet cloud as a platform, without the need to invest in several technologies in order to secure the communication within this virtual network, no additional hardware for creating VPN connection, and no third party solutions to automate the authentication process, or encryption or even depend on hardware routers to route messages. The ultimate goal is to establish a secured environment for data exchange, especially XML data files. In addition, to combine features from well known and widely spread protocols such as TCP and UDP into a system that can guarantee data delivery in a multicasting ability, along with an enhanced performance in terms of transmission speed, throughput, data security, and utilisation of the resources available. Section 4.1 gives an overview of the main characteristics for TCP and UDP protocols and how RIDX benefit from its features.

4.1 TCP & UDP PROTOCOLS OVERVIEW

TCP and UDP protocols occupy the major share of Internet, LAN and WAN networks, and are used as a transport layer for most implementations. Between TCP and UDP, RIDX comes as a common ground that combines characteristics from both protocols; combining the reliability of TCP connectivity with a high transfer speed of UDP, RIDX was built to benefit from TCP and UDP and to perform the best out of them. The next sections contain an overview of TCP and UDP protocols, and a description of RIDX UDP. Some major differences are

discussed between standard TCP and standard UDP, and then discuss the enhancements that have been made to construct and implement RIDX UDP.

4.1.1 Standard TCP

TCP is a highly developed and complex protocol. Nevertheless, although major improvements have been proposed and implemented over a period of time TCP has retained the most fundamental operations unchanged from the specification published in 1981 (RFC 793). The document 'Host Requirements for Internet Hosts' referenced at RFC-1122, specifies the amount of requirements for an implementation of the TCP-protocol. The 'TCP Congestion Control' in RFC-2581 is considered as the most significant document relating to TCP recently. It describes new techniques to pass up excessive congestion. Another document, written in 2001 (RFC-3168), described a new approach, which is 'Explicit Congestion Notification' (ECN), a form of mechanism to avoid congestion of signalling. In the early stages of the Internet, TCP occupied 95% of the total packets circulating on the Internet. Among the most familiar implementations that adopted TCP protocol were: e-mail family (POP3, SMTP and IMAP), Internet (HTTP and HTTPS), and other utilities such as Telnet and FTP. Its large extension has been tested for the original developers of its creation were outstanding.

Most recently, a new technique for controlling congestion was introduced and denoted as FAST-TCP which stands for 'Fast Active queue management Scalable Transmission Control Protocol' by developers from the California Institute of Technology. It detects congestion from delays in the queues experienced by packets to be sent to their destination. There is still an open debate on whether this is an appropriate symptom control congestion.

Operation Details:

TCP connections are composed of three steps: establishing a connection, transferring the data and lastly ending the connection. To connect using the procedure called negotiation in three steps (3-way handshake); disconnection used a four-step negotiation (4-way handshake). When establishing the connection, initialising some values of the parameters is done in order to guarantee the delivery of the data is in order and to guarantee the healthiness of the communication.

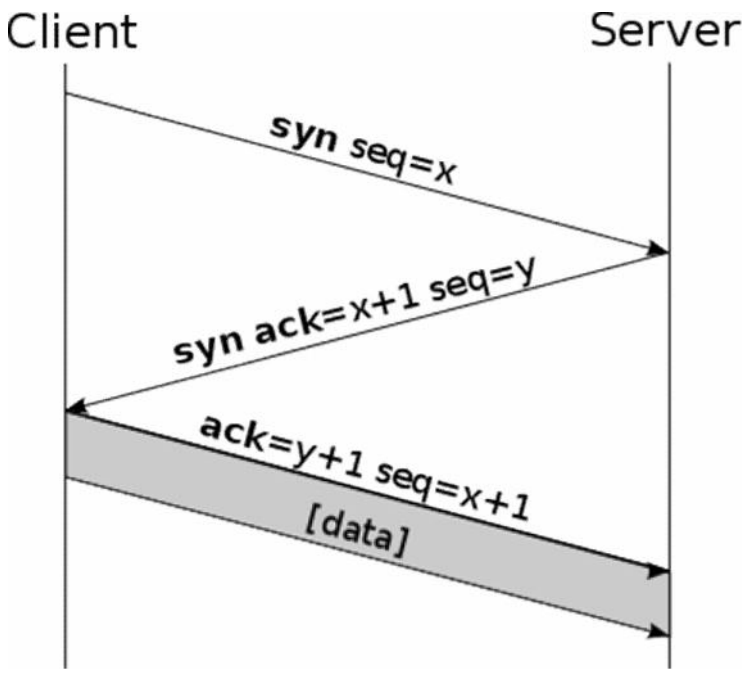

Figure 10: Trading in three steps or three-way handshake

Establishment of bargaining

When a sender tries to send data to a receiver, usually the receiver opens a socket on a particular TCP port, and this port listens for any data being sent by the sender. This is called a ‘passive open’, plus the connection determines the server side. The connection on the client side

performs an ‘active open’ port by sending a primary SYN packet to the receiver side (client) as part of the negotiation in three steps. The server side checks whether the port is open, i.e. if there is a process listening on that port. Should it not be, send to the client a response packet with the RST bit set, which means the rejection of the connection attempt. If we do open the port, the receiver side (server) would reply to the legitimate SYN request with a SYN/ACK. Lastly, the sender side (client) should reply with an ACK to the receiver (server), thus carrying out the negotiations in three steps (SYN, SYN/ACK and ACK) and establishing the connection step.

Sliding Window & Window Size:

The acknowledgments (ACKs or Acknowledgments) of data between the sender and the receiver can benefit from it; the sender especially can understand network conditions between them. When a timer is employed during the transmission, both the sender and the receiver can adjust the performance of the data flow. To accomplish high performance and circumvent any congestion in the network in TCP, there are some techniques used by TCP protocol for controlling the flow of data (the sender sends as fast as the receiver can receive). For instance, one of the commonly used techniques is ‘Sliding Window’, which controls the transmitter to send information within the receiver’s buffer and flow control algorithms, such as the algorithm for congestion avoidance (congestion avoidance), the slow start (Slow-Start), the fast retransmit, the fast recovery (Fast Recovery), and others.

The data bytes received is the window size that can be crammed into the receive buffer for the connection. The issuer may send a certain amount of data but you must wait for a nod of the window size update from the receiver.

An example would be: a receiver starts with a window size of x bytes received and then your window size is x - y and the transmitter can only send packets with a maximum size of data (x - y) bytes. The following packages received will reduce the size of the receive window. This situation will remain until the receiving application collects data from the receive buffer. For efficiency in networks of high bandwidth, you should use a larger window size.

End Connection:

The completion phase of the connection uses a negotiation in four steps (four-way handshake), ending the connection from each side independently. When one of the two sides wants to end his side of the connection, it sends a FIN request message and the other party nods with an acknowledgement. Consequently, a classic off requires a pair of FIN and Acknowledgement packets from both sides.

If one ended the connection but not the other, then the connection status is 'Half Open'. The side that has ended the connection cannot send more data but the other can still.

4.1.2 UDP Protocol

User Datagram Protocol is a transport level for exchanging datagrams. It allows sending datagrams through the network without having previously established a connection, since the same datagram includes enough addressing information in its header. Nor does it confirm or flow control, so that packages can anticipate each other, and it is not known if he has arrived correctly, and no delivery confirmation or receipt. Its primary use is for protocols such as DHCP, BOOTP, DNS and other protocols that exchange packets, or are not profitable with respect to the

information provided, as well as streaming audio and video in real time, where broadcasts are not possible because of the strict delay requirements you have in these cases.

Description:

The family of Internet protocols, UDP, presents a simple edge between the application and network layer. UDP does not provide any guarantee to deliver messages, and the source does not retain the states of the UDP messages that are sent to the network. It does the checksum on the data (payload) and the header. Any guarantees for transmission of the data should be done in a higher level.

Headers contain four fields from which two are optional. The 16-bit fields present the ports of the source and destination which identifies the origin and reception operation. Because UDP message is pertaining to state the message at the destination side, therefore the source may not know or ask for replies, and the source port can optionally be used, or else it should be reset to (0). The destination port field is mandatory and contains the destinations' port number. The data and the header length are calculated and set into the Length field, the 8-bytes length of the header in addition to the maximum 65,527-bytes length of data forms the maximum length number can be put in the length field, assuming that the data field contains no data, therefore the minimum length is 8-bytes. The 16-bit checksum field comprises the header data and a header with source and destination IP, protocol, the datagram length, and 0's to complete a multiple of 16, but not the data. According to IPv4, checksum field is optional, if not used it needs to be set to zeros, but as in the specification of IPv6 it does not.

The UDP protocol is used, for example, when you need to transmit voice or video and it is most important to speed to ensure that absolutely all the bytes arrive.

4.2 RIDX UDP PROTOCOL

RIDX UDP protocol is part of the transport layer for RIDX system. Transport layer consists of three main modules: RIDX UDP, ROUTER and TUNNEL respectively (see Figure 5). The following sections describe the main characteristics of each module with its functionality and major security threats to be considered especially on the OSI level. The common parts of this transport layer that are shared with other conventional protocols is the standardisation of port numbers and data transfer concept The following points describe further:

4.2.1 Port Numbers

Port numbers are used to distinguish between different applications used by the sender and the receiver. Both sides of the connected parties have linked to an unsigned 16-bit number (a possibility of 65,536 different port numbers) assigned by application on both the sender and the receiver. It is calculated by the total range of 2 to the power 16, and covers 65,536 numbers, from 0 to 65,535. The port classification defines three classes: Registered, Dynamic (private) and Well-Known. The latter ports are assigned by the ‘Internet Assigned Numbers Authority’ (IANA), ranging from 0-1023; these ports are commonly used by the operating system. Applications that use these ports are implemented as servers and are listening for connections. For instance: HTTP uses port number 80, SSH, FTP, Telnet, and SMTP uses consecutive port numbers 22, 21, 23 and 25. Users can connect to these ports on a temporary basis. The Registered port class represent services that are listed by a third party (registered port range: 1024 to 49,151). Dynamic ports/private class is to be used by user and application customisation. Dynamic-private ports have no meaning outside the RIDX UDP connection in which they were used (dynamic range of ports/private: 49,152 to 65,535). Rules for port numbers are implied on the transport protocols (TCP, UDP and RIDX UDP).

4.2.2 Data Transfer

Data stream through Transport layer to be sent to the network. The Shredder/Assembler module in RIDX divides and partitions the XML file vertically into a byte stream returned from the application into segments (chunks) of appropriate size and add headers. Then, RIDX UDP hands over the resulting segment to the IP layer, where through the network, it arrives at the RIDX UDP layer of the destination entity. Sequencer in RIDX assigns sequence numbers in order to check that no segment is missing, which is also used to ensure that the packets have reached the target entity in the correct order. Assent RIDX returns a byte that has been received correctly, a timer on the origin of the consignment entity states timeout status if the packet did not acknowledge that it received within a reasonable time, therefore, the probable missing packet will then be retransmitted. The Assembler module checks that no damage happened to the data packet during shipment, and it is calculated by the issuer in every packet before being sent, and checked by the receiver.

4.2.3 RIDX Message Structure

The main components comprised in RIDX message are the body which is compulsory and an optional header. The body is always aimed for the last message's recipient, whereas the interpretation of the message is processed according to header entry. Binary or other types of data can be attached to the message body.

Message header considers an orthogonal relation with the main or primary content of the message, therefore, it is a useful feature to be offered for the user, and it is so helpful in accumulating the data to the message that does not affect the message body's interpretation.

For instance, headers perhaps employed to give digital signatures for an appeal comprised in the message's body. Therefore for example, an authorisation or authentication service can use headers for this purpose, and strip out the data to authenticate the digital signature. And when the authentication is confirmed, the remaining part of the message will pass to the higher pack of modules and process the whole body message. A very close observation of the message will assist in making the function clear as well as positioning the header as well as the elements of the body.

The RIDX message structure is similar to IPv6 packet; XML data is shredded into smaller chunks, encrypted and encapsulated into RIDX message (Data Field), then the message can be sent to other domain members (Grötschel, Alevras and Wessäly, 1998). The message structure is as follows:

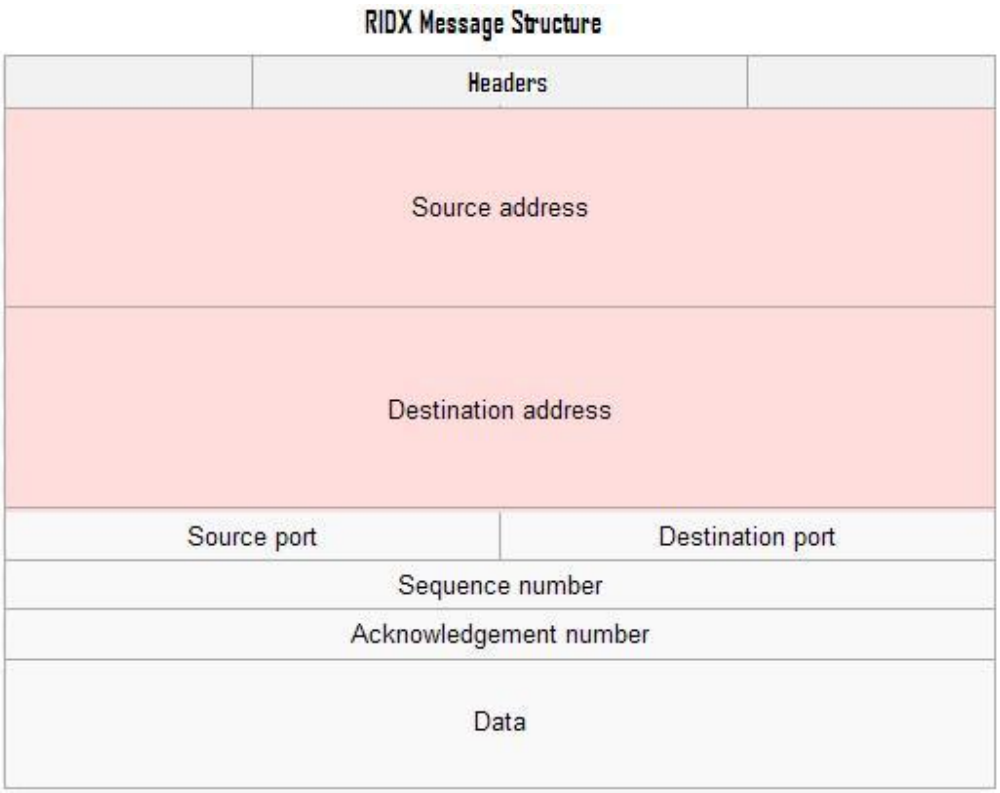

Figure 11: RIDX UDP message structure used in transport protocol layer

The RIDX message contains:

- **Message Headers**: different methods can be used to operate headers, for example, one method is responsible to attach/add headers to a message, another method is to detach/delete, and so on. Headers are attached whenever data should not be put in the Data Field. Headers indicate that both communicated RIDX nodes formerly concurred leading that the semantic who rule the interpretation of the header entity, with the intention that the service understands precisely what to perform with the elements' value. When the service obtaining the respond does not interpret the meanings of the header value, then the message will refuse the entire request and return an error. Therefore, this error response has to be as an element of the respond message body and forms a unique component, and should be defined as part of the response mechanism answering back the client if any message interpretation error happens. In addition to describing the interpretation of nodes as mentioned above, a RIDX message may in an optional manner comprise the header entries specifying nodes which execute the processing of authorisation, business logic processing, encryption, persistence of state, etc. Headers assist to develop RIDX as a modular, extensible model of packaging, taking into account that the body element of RIDX message and the header entity is processed independently.
- **Source/Destination Address:** usually the RIDX UDP transport protocol will fill the source address (Sender) automatically, but in some cases, the source wants the message responses to be delivered to a different address or other domain member.
- **Data Field:** it is the desired data to be transferred between domain members. RIDX message body comprises a payload devoid of not evident for hackers to realise and

understand. Therefore RIDX is not only a cryptic model of packaging, but also a modular packaging model. In the body, sniffers see the message as a combination of character set entity that has no meaningful sequence assuming that there is no encryption applied yet, this is due to the vertical fragmentation of the original XML file by the Shredder module; then it is up to the service and its configuration to interpret the meaning of the message and perform the correct response. Pack of modules effectively offers a clear form for handling the payloads in a meaningful and significant way. Here, the word ‘meaningful’ means the domain member requests to initiate procedures in order to invoke the pack of modules desired to interpret the payload of the message.

- **Source/Destination Ports:** the port number (Socket Number) that will differentiate between RIDX instances; the socket number will also work as Pipeline identifier.
- **Sequence Number:** the sequencer module will generate sequence numbers to the shredded/fragmented data (chunks) from the original XML file. This sequence number will preserve the data ordering when merged in the destination side. And the destination node will rearrange messages, as well as identifying each segment of the data (Peterson and Davie, 2000).
- **Acknowledgement Number:** RIDX is a Multicast reliable delivery, therefore message acknowledgement is to guarantee delivery of all messages, in case of lost packets/messages, and the source will retransmit the lost packets.

4.2.4 Protocols Comparison Chart

Differences and similarities between the three transport protocols TCP, UDP and RIDX UDP can be summarised in Table 2; this table recapitulates the main features for each protocol and the differences between them:

	TCP	UDP	RIDX UPD
Packet Ordering:	Orders of the data packets rearrange as specified.	No orders of the data packets. To preserve ordering, techniques should be added in a higher layer.	A sequencer module added to sequence the packets and manage the ordering
Error Checking:	Error checking is available	No option for error checking	Checking errors on a higher level
Headers:	Header is 20 bytes size	Header is 8 bytes size.	More headers can be added, the size of 8 bytes
Weight:	Sockets need 3 packets to set up the connection, afterwards data can be sent. Also responsible for congestions and reliability control.	Lightweight protocol, no tracking connections	Lightweight protocol, no need to track connections because connection is already established
Data Stream:	Interpret the data as a byte stream, bounded in a segment data size	Data Stream has no boundaries while packets has, data integrity checked only on arrival, packets are sent individually	Data Stream has no boundaries while packets has, data integrity checked only on arrival, packets are sent individually
Transmission speed:	Slower than UDP and RIDX UDP because of error-checking, 3 way handshaking…etc	Faster than TCP because there is no error-checking for packets	Result showed that it is faster than TCP protocol
Reliability of	An absolute guarantee for	No guarantee at all for the	A feature was added (Ack-

	TCP	UDP	RIDX UPD
data:	the data to reach destination	data to reach destination	Nack) & sequencer to guarantee delivery and order
Operational:	3 way handshaking or 4 way handshaking to establish a connection	No need to establish a connection	Connection already established in step ahead by Pipeline registration
Reliability of connection:	Reliable on Two way Connection	Reliable on one way Connection	Reliable on Multi Way Connection
Flow Control:	Contains Flow Control technique	No option for data flow control	A Data Flow Control (DFC) module was added

Table 2: Protocols comparison chart

The RIDX UDP transport layer employs IP Multicast with added features to become a Multicast Reliable Transport Protocol transmission functionality. In addition it is a reliable transmission by adopting a ACK-NACK (Acknowledgment & Negative Acknowledgment) message control mechanism, which provides as an option a 'first in first out' (FIFO) technique for guaranteed message delivery transmission. The summary in Table 3 concluded the differences between protocols and shows differences in terms of reliability. In Unicast and Multicast transmissions, the row header represents both unreliable and reliable model, and the column header represents the Unicast and Multicast transmission. Therefore, items in each cell in the table meet both the delivery mode and the transmission mode criteria together, for example, TCP protocol lie under Reliable and Unicast, which satisfies both characteristics, and RIDX UDP satisfies both Reliable and Multicast transmissions:

	Unreliable Delivery	***Reliable Delivery***
Unicast Transmission	U D P	T C P
Multicast Transmission	IP Multi-cast	RIDX UDP

Table 3: Summary of protocols differences

In order to define what Reliable Delivery Transmission is, we need to understand what unreliable delivery is; for example in Table 3, standard UDP is categorised as unreliable delivery transmission, because of the following two main reasons:

- Messages can be dropped during transmission because the size of the message is too big, and no fragmentation mechanism is invoked to control message size, or the buffer at the receiver has reached an overflow state, then the receiver starts dropping messages. Or the switch buffer (NIC/IP network buffer) can also reach an overflow state and start dropping messages.

- No guarantees at the receiver side to receive messages in sequence, because UDP sends messages without acknowledgement mechanism, which means that if any message is dropped during transmission, the receiver will not request any dropped messages from the sender to be resent. Furthermore, the sender does not know the state of the message whether received or not.

4.3 RIDX ROUTER

In RIDX domain, a group of RIDX instances that share the same domain name will form a domain group. RIDX conducts and maintains the communication between its members. The connectivity between domain members is governed by the domain coordinator through a periodic and continuous check up on the connection between all members. On the other hand, each RIDX instance checks and maintains the connectivity between other instances using a PINGER module; the latter is similar in functionality to the Ping utility used in TCP protocol. The coordinator and all other members schedule timed intervals to issue Ping packets in order to check if the connection is 'Alive'. In case of a crashed member or any lost connection to a node, the coordinator announces to all active members that a certain node has crashed, and drops the node from the active list, then distributes the list again to all members.

The Active Member List is a crucial element to be used by the RIDX router, it is considered as a routing table to the routing module. Routing in RIDX is considered to be a single delivery to a specific node and a Unicast forwarding message to the target machine or instance, and resides between the sender and the receiver, plus the routing node performs as an intermediate member for forwarding messages from the sender to the ultimate destination. RIDX routing module is designed to route messages only for a one level depth or distance, and the router can only route a message to the final destination and not to another RIDX router. For example, if a domain contains four domain members (A, B, C and D), and member A wants to send a message to D, the possibility in our case here has three possibilities to send from A to D, one by sending messages directly from A to D, or sending from A through B to D, or sending from A through C to D. Node B cannot route the message to C through D.

Standard UDP and RIDX UDP share the same characteristics in terms of operational transmission; both protocols do not have a dialogue for implicit handshaking model as in the case of TCP protocol, RIDX rather uses customised modules at the application level. This technique can save the operational processing time at the communication level and avoid the overhead of requiring special paths or prior communication setup to transmit data. Therefore, setting up the communication ground between all RIDX domain members prior to transmission time can save communication processing time, improve the transmission speed and as a result improve the overall performance of data interchange. Section 4.1.2 showed that UDP protocol is considered as a non-reliable protocol which means no guarantees to deliver data and no consideration of data integrity or message ordering. The RIDX system has well thought-out these shortcomings by adding modules to support the transport layer and modules that the transport layer is set on top of , such as routing and tunnelling modules (see Figure 5).

The RIDX Router module is a simple yet effective technique to forward messages. It simply reads and interprets a stateless nature of RIDX UDP message by reading the Source/Port and Destination/Port addresses together with the headers of the message (see Figure 11). If the destination address does not match the routers' address, then the router looks up into the active member list to match the address with one of the addresses within its list, and if it exists, it directly forward the message to its destination. If the address does not exist then it will check the headers for a specific code parameter or any attached instructions. In certain cases some instructions or parameters can be attached to the message in order to pass these instructions to other members. For example, when one of the members resides behind a firewall and the group members need to maintain communication with the hidden member through a dedicated router; this case is discussed more in section 4.5 when applying the tunnelling technique.

4.4 OSI & SECURITY THREATS

The OSI model contains different layers, and each one has its own security risks, which developers are working on overcoming. The physical layer is a layer that must be approached from a physical point of view, because access to this layer is most likely to come from outside the device level. The threats to the physical layer include people taking the equipment itself, packet sniffing, tapping, electric failure or disaster damage.

These threats can be combated by using identifying badges, locks and surveillance equipment to reduce the exposure of outsiders to the equipment as well as sniffer equipment to identify leaks on the cabling. To protect against disasters, electromagnetic shielding and distributed data backups may be used along with backup power supply.

Establishing the identity by knowing the MAC address occurs on the data link layer; at this layer, spoofing this address can occur at the upper layer. Additionally, faults when introduced can cause loops with spanning tree protocols. There is vulnerability at this layer with the use of virtual LANs, where the interconnection of LANs and wireless LANs with VLAN policies can be used to perform VLAN hopping, creating data pathways to bypass firewalls and subnet addressing.

These threats can be mitigated by separating the sensitive areas from the rest of the physical network and by using security at other layers to establish security for VLANs. Wireless networks must be secured and unauthorised wireless access points should be detected and removed as soon as they are discovered. VPNs also provide a level of security at this layer by allowing encrypted data transfer. Network Intrusion Detection (NID) systems can be implemented here to watch the data and look for suspicious packets.

At a third layer, security controls are a challenge for both Internet and routing protocols. Security threats can be used such as route and address spoofing, using a machine's own address to send out malicious packets that appear to be from within the network. Firewalls that can be configured at the edge of a network to closely examine packets coming in from outside a network will reduce the chances of these kinds of attacks. Routing controls and filters should be used alongside ARP monitoring software. IPSec will also prevent insertion at the transport layer of any packets.

Error checking occurs at the transport layer and uses UDP or TCP protocols to route packets around the network and ensure successful delivery of data. When the transport layer receives packets, which are not well defined, some protocols have difficulty handling those packets. This is the layer where Denial of Service (DOS) and DDOS attacks occur (Stanton, 2005).

The solution to problems at this layer is the use of firewalls with stateful packet inspection and dynamic NAT (Arizona Enterprise Architecture, 2005). Some controls used to filter unwanted packets to prevent the DOS attacks; these controls can use software or hardware devices such as firewalls to do so (Arizona Enterprise Architecture, 2005). Part of the security from this layer involves the methods used in the session layer to implement encryption and prevent man in the middle attacks; SSL and SSH also come into play here.

The session layer begins, manages and ends dialogues between devices or applications (Song, 2004). At this layer, application program interfaces (APIs) such as NetBIOS and remote procedure calls (RPCs) allow communication among the upper layer applications. It is also at this layer that problems with authentication and session identification can occur, as can brute

force attacks and information leakage. It is feasible to encrypt passwords alongside SSL, SSH or transport layer security (TLS), with wireless TLS when required, to provide greater security. Expirations and timeouts can also improve security at the session layer (Mazurczyk and Szczypiorski, 2009).

The presentation layer handles the compression, encryption and standardisation of the data for the application layer to the session layer to remove differences in the format of data. Unicode vulnerabilities can allow users access to the root directory when it is kept in the same place as the files for the Internet. Buffer overflows are also an issue in the presentation layer, when temporary storage areas are flooded with more information than the interface can handle. The vulnerabilities of input weaknesses can be limited by careful coding and testing interfaces before implementation (Stanton, 2005).

The application layer is perhaps the most difficult to keep secure because of the wide variety of applications in use. Back doors, security flaws, which are inherent or written unintentionally into the application code, and new threats that were not apparent when the application was written, are all factors at this layer. Trojans, viruses, worms, spyware attacks and password issues all come into play at the application layer. The use of strong passwords and enforcement of password policies can reduce issues with access, and application firewalls can reduce the misuse of applications. Digital certificates using PKI and encryption can also assist in network security. Additionally, anti-virus software and anti-malware programs can keep machines free of Trojans, adware and spyware (Mazurczyk and Szczypiorski, 2009). SQL injections, cross-site scripting and parameter tampering are three very common application layer attacks which require a new way of approaching security. In some cases, there is a need for testing of the application

before it is put into production, but in other cases, companies are justified in bringing in outside companies to test for weaknesses in the network's application layer (Stanton, 2005).

In building any network, there are many things to be aware of that can create vulnerabilities in the system, but with research and persistence, a network administrator can plan for most instances of failure and keep the majority of the network and its data safe. Backups of data and a good knowledge of the physical and virtual layout of the network are imperative for keeping most of the network assets from becoming vulnerable to attack from inside the company or outside (Fielding, 2007).

4.5 TUNNELLING

Firewalls stop any attempt to contact an internal method from outside the corporation through the Internet on other open ports (Stanton, 2005). Firewalls will moreover evade users of internal networks from communicating with particular Internet sites which could be offensive or risky (Shay, 2004). The firewall performs at the OSI model's layers three and four through looking packets for particular kinds of headers. So, firewalls vary from file safety as anyone inside the corporation can supposedly contact files behind the firewall, whereas security of files presents internal safety against staff of a corporation (Chaudhury, 2005).

One of the obstacles that can face RIDX is conducting proper communication when an RIDX instance resides behind a firewall. Due to a great risk in using public Internet communication, organisations use a firewall to protect the organisation against attacks, and impose access control. For that reason, firewalls mostly prevent the connection from outside to inside, but allow some inside applications to connect to outside services and machines.

Tunnelling is a protocol that encapsulates other protocols, and provides a mechanism to allow RIDX to operate through firewall systems and security gateways:

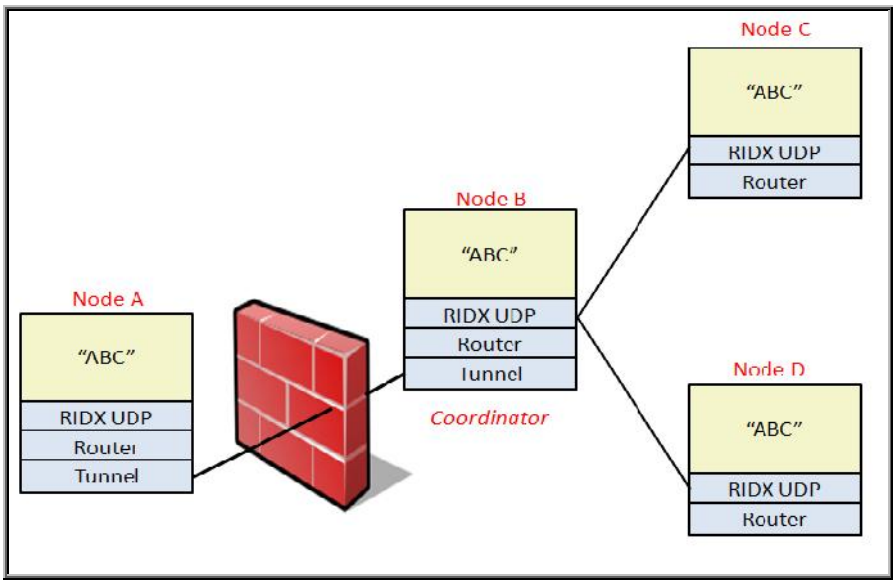

Figure 12: Creating a tunnel between a node behind the firewall and a node in the Internet cloud

The question we need to ask is: "Is it feasible to implement VPN tunnelling between all RIDX instances using external or third party hardware and software?" In order to answer this question, we need to look at different aspects when applying it.

First, implementing tunnelling can be expensive mostly due to the need for different technologies involved, such as hardware, software and security network specialists to implement and maintain it (Leonov, 2004).

Second, when hosting RIDX instances internally in the intranet organisation, tunnelling is not applied, and still the transferred data can be exposed by man-in-the-hub.

Third, when using embedded tunnelling mechanism within RIDX modules, performance is a considerable factor that needs to be investigated. Therefore, an experiment has been made to measure the throughput before and after a VPN tunnelling on different data frame sizes. More details about the experiment test results are discussed in chapter 5.

Considering the communication structure of RIDX, which depends on IP networking (multi-casting), RIDX has followed the recommendation stated by the Internet Engineering Task Force organisation (IETF) standards by using Layer-2 Tunnel Protocol (L2TP) over IP networking. The second available option for tunnelling is widely used especially in a Windows environment such as point-to-point tunnelling technique (PPTP), but the latter does not comply with IETF standards, and the encryption method which is used for this tunnelling type is not considered strong enough compared with Internet Protocol Security (IPSec) encryption method used with L2TP (Wanger et al, 2006). One of the two main schemes in IPSec is encapsulating the message using ESP, which stands for ‘Encapsulating Security Payload’. Most implementations of IPSec use ESP as a base module, as it is considered to be more secure than the other IPSec scheme pointed as Authentication Header (Arturo Perez et al, 2006). Therefore, RIDX has adopted the L2TP/IPSec/ESP methods for tunnelling through firewalls and Internet cloud. A test has been made to measure the performance of tunnelling the connection over Layer-2 Tunnelling Protocol using IPSec/ESP scheme, and the results are as shown in chapter 5.1.

CHAPTER 5

5. Experimental Results & Case Studies

5.1 RIDX TUNNELLING PERFORMANCE TEST

In section 4.5, Tunnelling was presented as part of RIDX modules and a technique to create a secured connectivity between RIDX domain members as well as how it can be applied in different situations. For example one case is to penetrate the firewall, or in some cases, through an un-trusted Internet cloud, or maybe across shared infrastructure network connectivity. The purpose of tunnelling is to create secured layer between two RIDX systems or multi tunnelling connection between domain group members. A test is made to measure two things:

1- The throughput before and after tunnelling in Megabit per second for different frame sizes.
2- The Latency before and after tunnelling in microsecond for different frame sizes.

The test is conducted on SUN 3 GHz with 4GB RAM on 2.6.9-14 RedHat Enterprise Linux 4 / 64 bit system that the kernel natively include and support the IPSec, and the tunnelling passes through a CISCO switch which is 1GBps along with Gigabit Ethernets installed on RIDX nodes, a SUN JVM 1.6.0_3 is installed on Two nodes which contain RIDX system. The tunnelling connection was established between two nodes, and all messages were sent in a unidirectional way (only one way) from node A to node B, and different frame sizes used, the frame sizes varied from 64 bytes up to 1024 bytes. Table 4 shows the experimental results of measuring the

throughput before and after tunnelling on each frame size used. Traffic generated at 100% of the line rate of (1000 Mbps).

Frame Size	Throughput BEFORE Tunnelling	Throughput AFTER Tunnelling
64-bytes	34.4 Mbps	7.1 Mbps
128 bytes	68.7 Mbps	12.3 Mbps
256 bytes	137.3 Mbps	19.2 Mbps
512 bytes	210.4 Mbps	26.8 Mbps
1024 bytes	190.5 Mbps	33.5 Mbps

Table 4: Throughput Data of VPN (Before and after)

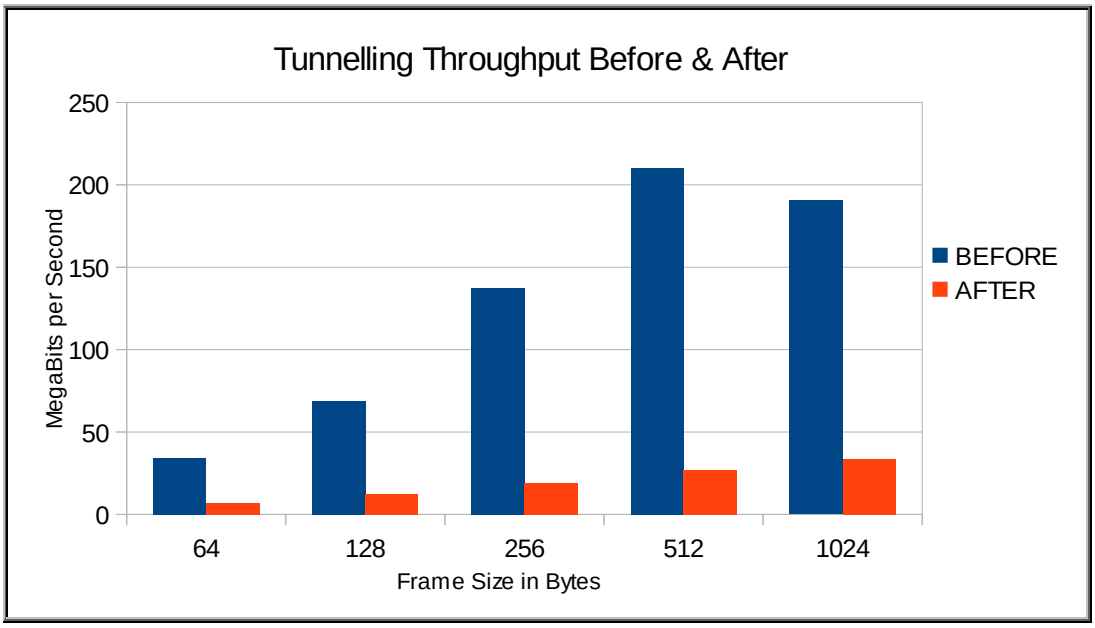

Figure 13: Tunnelling Throughput Before and after

The results in Table (4) & Figure (13) showed that there is a significant effect on the throughput when using tunnelling techniques over IP networking, The effect of tunnelling varied in relation to the frame size, for example, on 64 bytes frame size, the throughput has dropped down by 79% approximately, and on 1024 bytes frame size, the throughput has dropped down by around 82%. The significant effect was noticed on the 512 bytes frame size which dropped down the throughput by 87% and the second highest throughput drop down is on 256 bytes frame size.

The delay of delivering the packets from node A to node B is measured before and after tunnelling, the tunnelling test is applied under the same setup conducted above, time is measured in microseconds for 1 packet traffic per second. Different frame sizes are used and varied from 64 bytes up to 1450 bytes. Table 5 shows the result of this test:

Frame Size	Delay BEFORE VPN tunnel (µs)	Delay AFTER VPN tunnel (µs)
64 bytes	165 µs	315 µs
128 bytes	169 µs	334 µs
256 bytes	171 µs	374 µs
512 bytes	196 µs	445 µs
1024 bytes	228 µs	612 µs
1450 bytes	260 µs	736 µs

Table 5: before and after VPN one-way tunneling

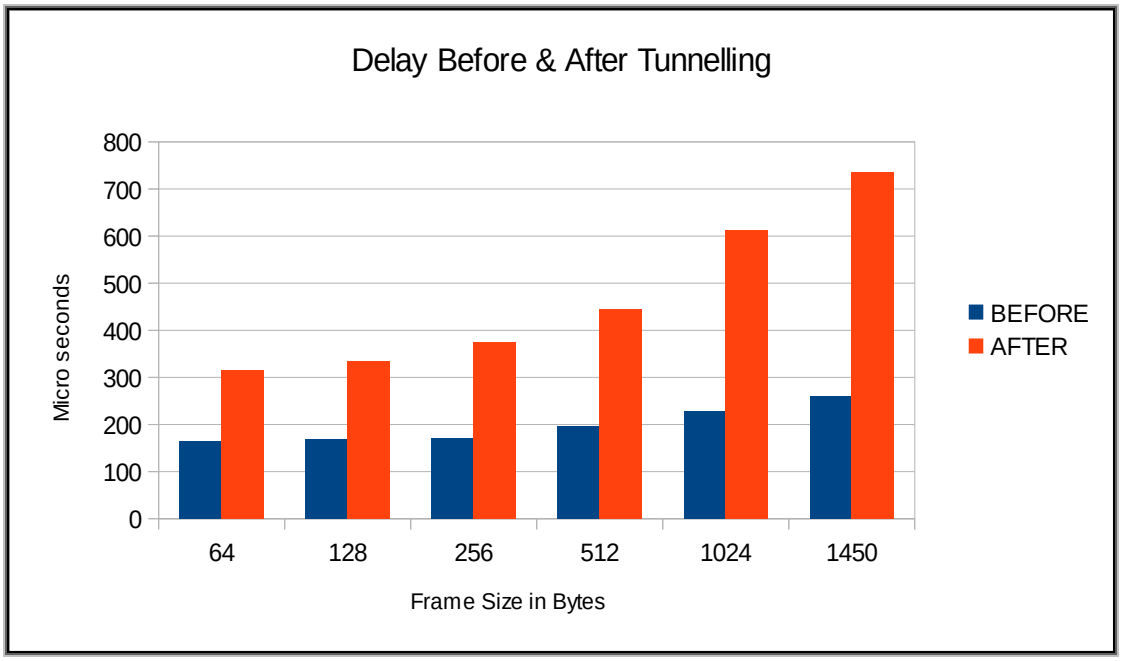

Figure 14: Delay before & after tunnelling

The results showed that there is a noticeable effect on the packet delay time when conducting the tunnelling techniques, the delay time fluctuates depending on the frame size used, for example a significant jump in delay time by almost 3 times before and after tunnelling, was noticed when using 1024 bytes frame size and above.

As a result of this experiment, using tunnelling technique between all RIDX nodes needs a significant amount of overhead processing, although the results showed that there is considerable latency when using tunnelling techniques, consequently, choosing tunnelling in certain situations is acceptable where RIDX resides behind a firewall and has limited access in and outside the RIDX cloud. A module TUNNEL is responsible for this process. In all cases, there must be at least two ports open through firewall inside-out to be able to connect and conduct the tunnelling.

As in figure (12), the initial process of conducting a secured tunnel starts from (Node A) which is assumed to be behind the firewall by requesting to join a domain from the coordinator, the coordinator accepts the new joiner, Node A establishes the secured connection after the coordinator accepts it, then (Node B) the coordinator announces that (Node A) is only accessible by (Node B). Eventually, if any RIDX node in the cloud other than Node B tries to ping (using the Pinger Module) Node A, it will return a negative response and will fail, therefore Node B will act as a router for all messages to/from Node A.

Nevertheless, tunnelling technique in RIDX has several downsides. First, using Node B as a dedicated router might exhaust and consume the resources from Node B. second, It isn't always possible to control firewalls owned by a third party to setup and enable the desired ports needed to establish the tunnelling.

5.2 RIDX PERFORMANCE TEST PARAMETERS

A test is conducted on SUN 3 GHz with 4GB RAM on 2.6.9-14 RedHat Enterprise Linux 4 / 64 bit system, a CISCO switch of 1GBps, SUN JVM 1.6.0_3. Each node (box) contains RIDX system, it sends number of M messages to all cloud nodes; receiver can be a sender at the same time, the timer starts when first message is received, again the timer will stop when last message is received, number (M) and size (S) of messages is known to all RIDX nodes. The computation of message rate in each RIDX with its throughput is calculated by multiplying {number of senders N} times {number of messages M}. A configuration file contains same parameters for all RIDX nodes:

- N = 4 then 6 then 8, senders are same receivers, that means N equals to number of RIDX nodes also

- M = 1,000,000
- S = 1 then 2.5 then 5 KB the size of each message

5.3 EXPERIMENTAL RESULTS

The computation of message rate = (M * N) / T, where T is the total time of receiving all messages. Therefore, X & Y axis represents (S) and (Rate & throughput) simultaneously. For example: a collection of data results when the test is done, we need to calculate the average rate of number of messages received per second, this can be done by summing up the data collected and divide it by (N), then we calculate the throughput by multiplying the (S * average message rate) (FIX Protocol Ltd, 2009) .

When compared standard TCP against RIDX UDP, a TCP & UDP variance between four to ten nodes, we notice that the TCP begins at 49 MB per Second until it drops down to 34 MB per second, on the other hand, RIDX UDP begins at 56 MB per Second until it drops down to 51 MB per second, this shows that a significant better performance accomplished by RIDX UDP. But we also notice that when S = 2.5k, UDP drops down from 94 to 65 MB per second, and the same for larger message size; this is due to Maximum Transmission Unit system OS kernel parameter limitation to size of 1500, so whenever message size gets large and exceeds the Maximum Transmission Unit size, then the UDP packet split into more IP packets, , reasoning more delays to following packets, this problem we do not find it in TCP protocol because it creates IP packets which is always under than Maximum Transmission Unit size, and Shredder/Joiner modules is responsible of keeping the MTU size controlled by RIDX.

5.3.1 Case Study 1: Four Domain Members

This test is conducted on 4 RIDX domain members, Node A, B, C and D. Each node is capable of sending, receiving and routing messages at the same time; therefore, each node will send 4000000 messages and will receive 4000000 messages in total. For example, Node A will send 1000000 messages to node B; in this case node C and D will act as routers. And then node A will send 1000000 messages to node C; in this case node B and D will act as routers, and so on. While node A sends messages, Node B is also sending 1000000 messages to node A and in this case node C and D acts as routers, in the same sense, node D sends 1000000 messages to node C while node A and B will act as routers, and so on. The last considered case is when node A for example sends 1000000 messages to itself (node A to node A), in this situation the nodes B, C and D will act as routers. The loopback is not used in any test, and the sender must use other nodes to route back the messages to and from the sender.

The test was conducted on three different message sizes. The first message size is 1K, which is less than the MTU size, as mentioned in the previous section, the 1K size in this test is chosen to eliminate the possibility of fragmenting any messages which is over the MTU size (1.5K). Then the 2.5K and 5K message sizes were chosen to test the effect of the message fragmentation done by the operating system on the performance of the RIDX system, also to measure the effect of the use of the node resources (such as CPU time, Memory...etc).

The following next two graphs illustrate the test results, one for measuring the message rate which shows the number of sending messages per second per message size, and one for measuring the throughput rate which represents the capability of transmitting Megabits per

second per message size. The results below show a comparison between RIDX UDP and standard TCP under the same configuration environment setup:

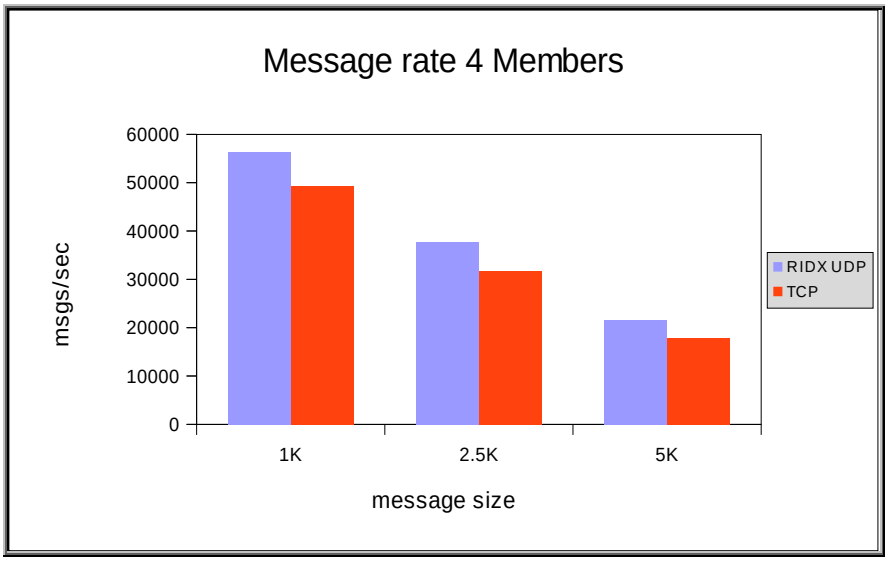

Figure 15: Message Rate for 4 Members

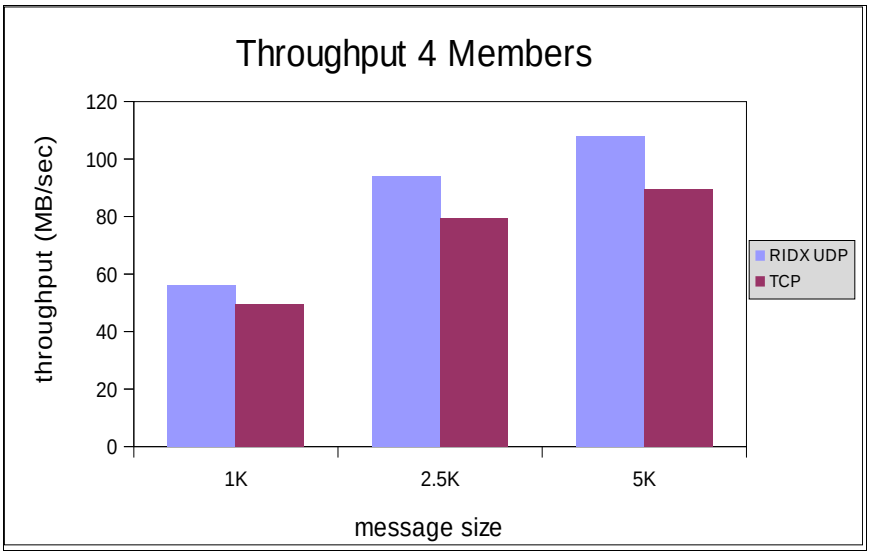

Figure 16: Throughput for 4 Members

Figure 15 shows a comparison of the message rates between RIDX UDP and TCP for various message sizes. A significant improvement is noticed when using RIDX UDP over TCP.

For example, on 1K message size, RIDX sent about 56 thousand message per second, whereas TCP sent about 49 thousand message per second. On 2.5K message size, RIDX sent about 37 thousand message per second compared to 31 thousand message per second sent by TCP, The last comparison based on 5K message size again showed better rating for the RIDX UDP over TCP with 21 & 17 thousand messages per second consequently.

Figure 16 shows a comparison of the message throughput between RIDX UDP and TCP for various message sizes. A significant improvement is noticeable using RIDX UDP over TCP on all message sizes. The throughput for example on 5K message size in TCP has achieved around 89MB/Sec while RIDX UDP has exceeded 107MB/Sec. On 2.5 message size, TCP has achieved around 79MB/Sec while RIDX has achieved around 94MB/Sec. And on 1K message size, TCP reached to 49MB/Sec while RIDX reached to around 56MB/Sec.

5.3.2 Case Study 2: Six Domain Members

The test parameters and the setup environment is the same setup explained in section 5.2, and these settings were used in all case studies that measure the message rate and the throughput. Figures 17 and 18 shows the comparison results between RIDX UDP and TCP, it shows the comparison of message rate and the throughput comparison respectively, In this case study, 6 members have formed the domain and established a connectivity between each other. Every domain member holds a list of other domain members in a form of paired field list which contains the coordinator IP address and the IP addresses/Port Numbers for all members. Each member will send 1000000 messages to every member in the domain; therefore, each member will send 6000000 messages and receive 6000000 messages in total including the loopback

messages, as explained in the case study 1, the loopback here means that the node will send messages to itself through the domain members and not using the internal loopback.

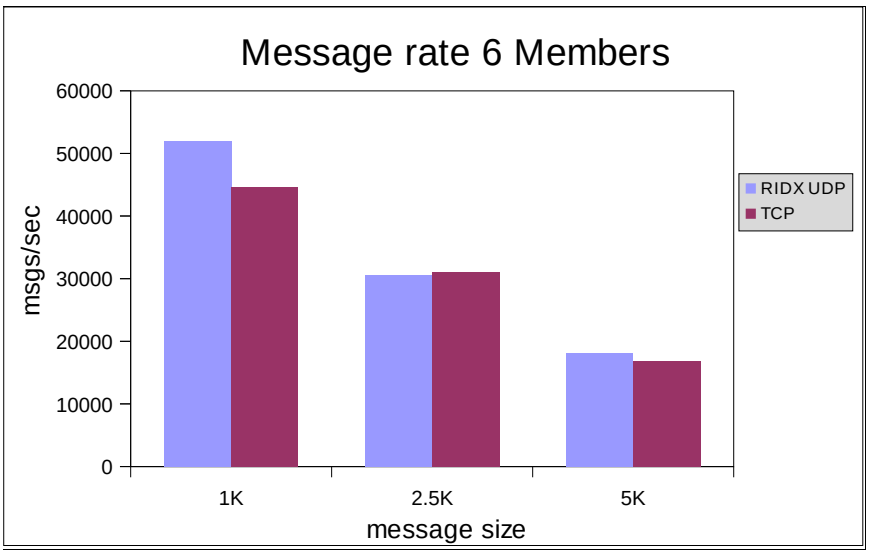

Figure 17: Message Rate for 6 Members

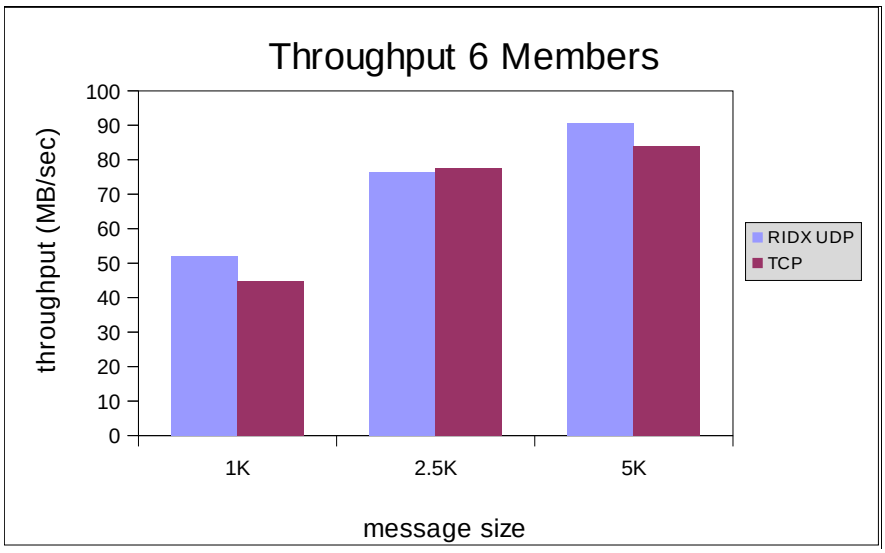

Figure 18: Throughput for 6 Members

Figure 17 shows a significant improvement using RIDX UDP over TCP on a 1K message size, RIDX has sent about 52 thousand message per second, whereas TCP sent about 44

thousand message per second. On the other hand, the 2.5K and 5K message sizes has ranged nearly closed to each other when comparing the message rate between RIDX and TCP.

In Figure 18, the results show that there is a significant improvement using RIDX UDP over TCP on both 1K and 5K message sizes. But the throughput on 2.5K message size on both RIDX and TCP has achieved almost the same rate which varied between 76 and 77 MB/Sec.

5.3.3 Case Study 3: Eight Domain Members

In this case study, the same test parameters have been used as in the previous case study, but adding another two members to the domain to become a total of 8 members in the same domain, and to study the effect of number of the domain members on message rate and the throughput. Every member will send, receive and route simultaneously 1000000 messages to/from each domain member, the total messages would be 8000000 per node. Using the multicasting techniques via multi threading pool of IP multicast, this gave the ability to send messages from one member simultaneously to all other members, and receive messages from all members at the same instance, and also routes all the messages that are not targeted the node.

Figure 19 shows the message rate comparison test results between RIDX and TCP on different message sizes

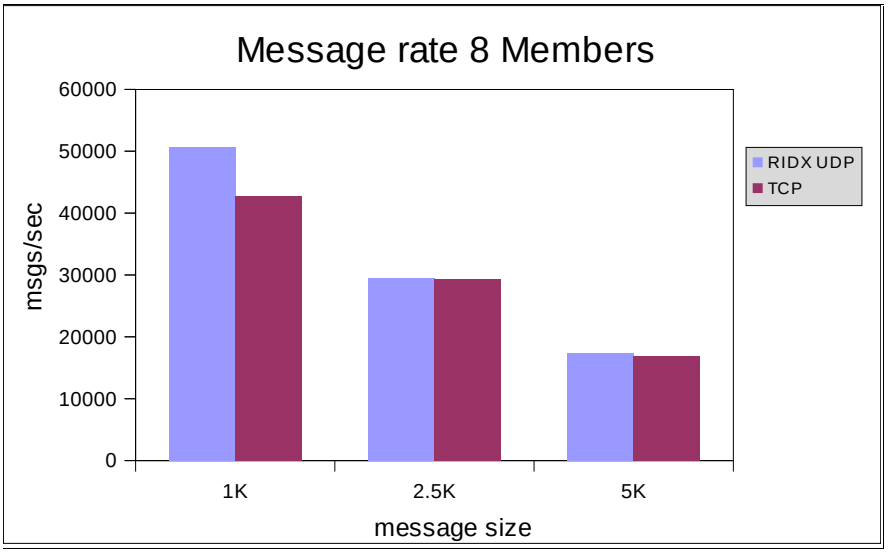

Figure 19: Message Rate for 8 Members

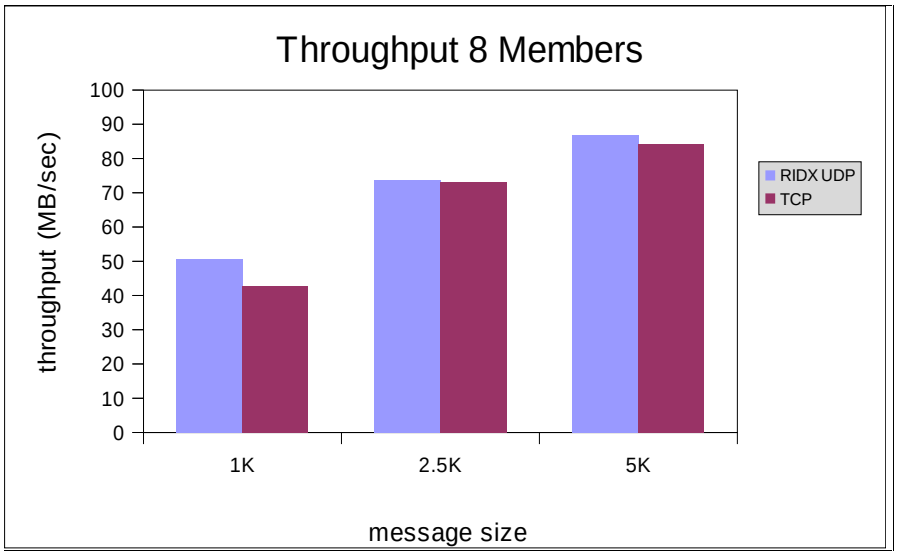

Figure 20: Throughput for 8 Members

The results showed in figure 19 that there is a significant improvement using RIDX UDP over TCP on a 1K message size, RIDX was capable of sending around 50 thousand messages per second, whereas TCP sent around 42 thousand messages per second. While the 2.5K and 5K message sizes has achieved almost the same message rate.

In Figure 20, the results showed that there is also a significant improvement on throughput using RIDX UDP over TCP on 1K message size. The 2.5K and 5K message sizes both achieved almost the same throughput rate with a slight improvement on 5K message size to RIDX UDP (86MB/Sec to RIDX against 84MB/Sec to TCP).

5.3.4 Case Study 4: Ten Domain Members

This last case study has used 10 domain members and joined in a single domain. The configuration setup and the environment parameters were preserved for consistency purposes and used the same setup as in the previous case studies. The connection between all members was established and by default, the coordinator governs this process. As in previous case studies, each node is going to send 1000000 messages for each domain member including itself, therefore each member will send in total 10,000,000 messages and receive the same. At any instance, each node can send, receive and route messages to/from all members at the same time. Figures 21 & 22 shows the results for message rate and throughput of this test:

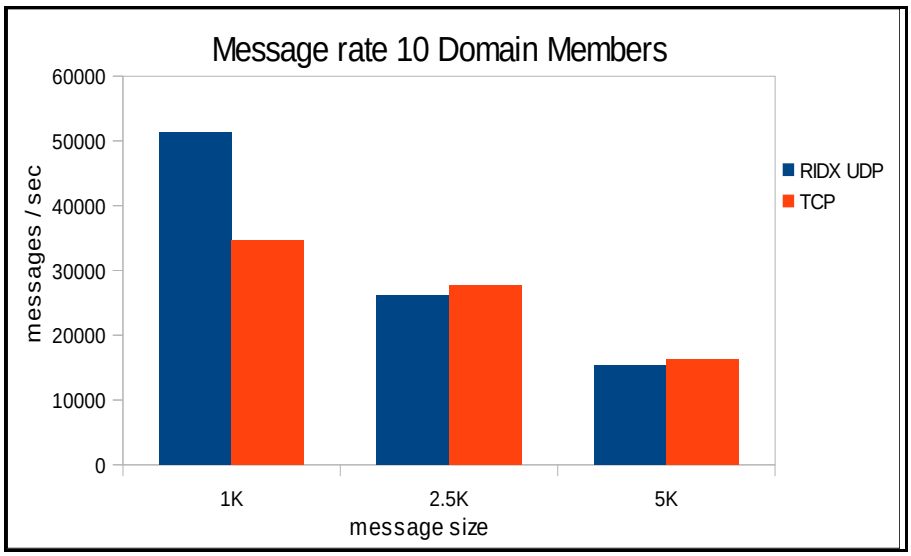

Figure 21: Message Rate for 10 Members

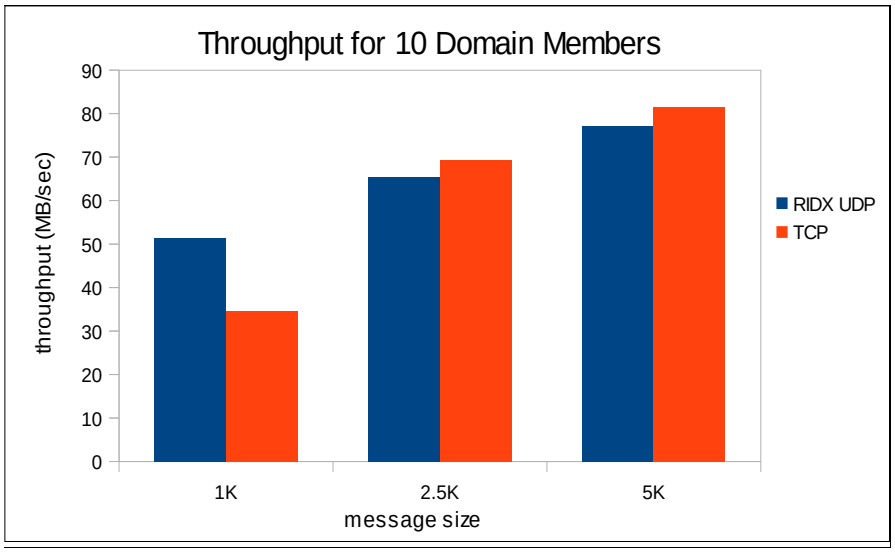

Figure 22: Throughput for 10 Members

Figure 21 shows a significant improvement using RIDX UDP over TCP on a 1K message size, RIDX has sent about 51 thousand message per second, whereas TCP sent about 34 thousand message per second. On the other hand, the 2.5K and 5K message sizes has ranged nearly closed to each other when comparing the message rate between RIDX and TCP.

In Figure 22, the results show that there is a significant improvement using RIDX UDP over TCP, RIDX throughput has achieved around 51MB/Sec whereas TCP throughput has achieved around 34MB/Sec. But the throughput on 2.5K and 5K message sizes on both RIDX and TCP has achieved almost the same rate which varied between 65 and 81 MB/Sec.

5.4 TEST RESULTS DISCUSSION

Many factors could affect the results when conducting the experiments, some of these factors have a direct effect, and other factors have an indirect effect. In this section, the author wanted to shed light on some of these factors, and also study the consequence of the experiment

test on the resources used during these tests, such as CPU time, Memory usage...etc. In section 5.2, the configuration parameters were set to be as a fixed setup for experimental tests for all the case studies conducted. The purpose of setting up the same configuration parameters and the same environment setup is to study the effect of the results when expanding the domain by adding members to the domain in terms of message rate, throughput, memory usage and CPU processing time, and to pinpoint the important parameters that might have an influence of the results. By identifying these parameters, it can be adjusted to enhance the performance and achieve better results.

5.4.1 A Comparison between the Case Studies results

Figure 23 compares RIDX against TCP on message rates between the 4 case studies (5.3.1 to 5.3.4) above, and groups them according to number of domain members and the size of the message. Each bar represents the number of messages sent per second per domain member. The X axis represents the number of domain group (between 4 to 10 members per domain), and the Y axis represents the number of messages scaled in 10000 messages.

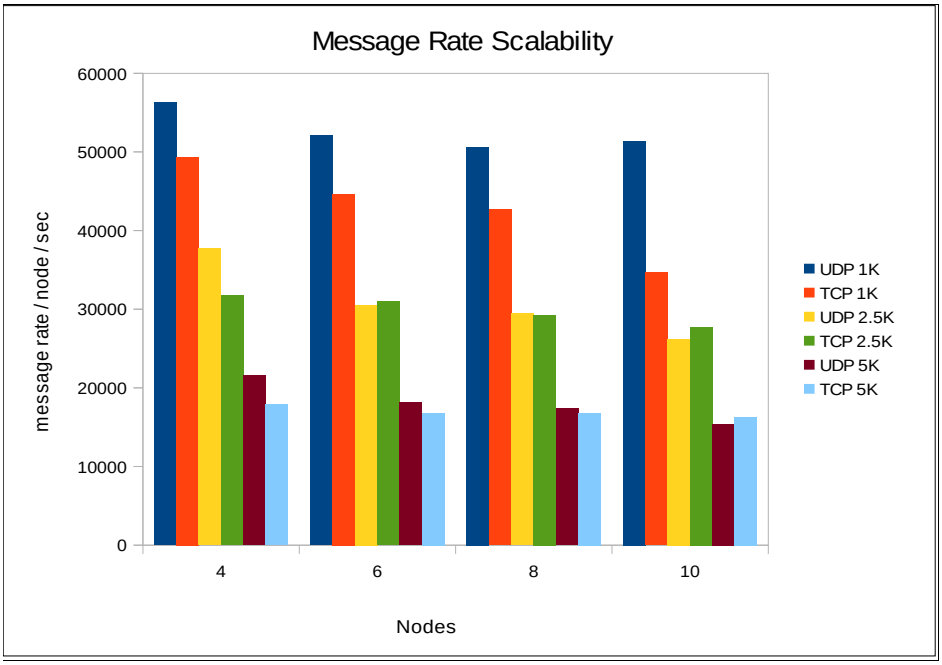

Figure 23: Message Rate Scalability

Figure 24 compares RIDX against TCP on message throughput between the 4 case studies (5.3.1 to 5.3.4) above, and groups them according to number of domain members and the size of the message. Each bar represents the Megabits sent per second per domain member. The X axis represents the number of domain group (between 4 to 10 members per domain), and the Y axis represents the throughput in Megabits per second scaled in 20MB/sec.

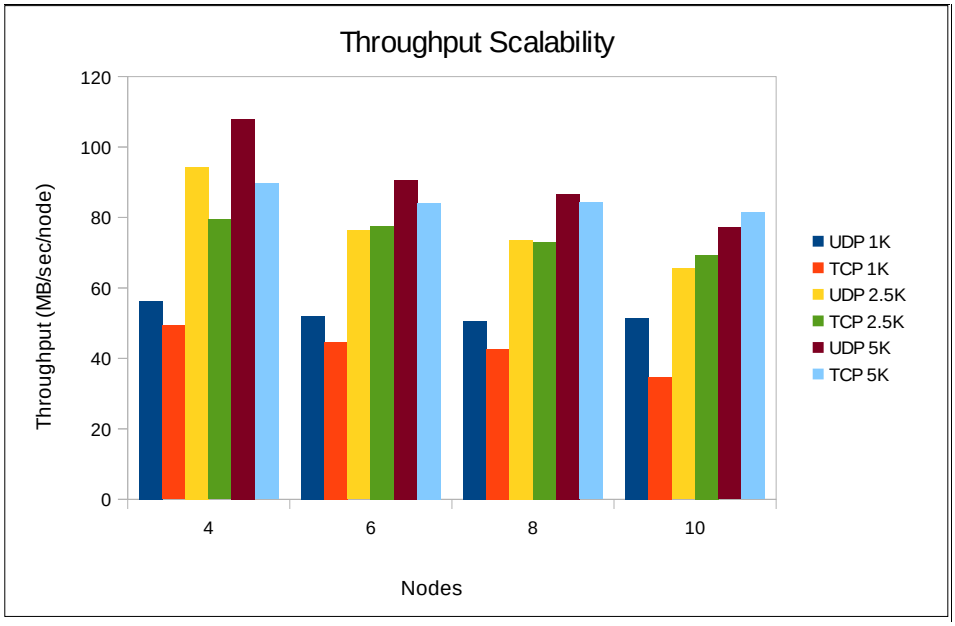

Figure 24: Throughput Scalability

The comparison graphs in Figures 23 & 24 showed that there was a slight affect in both the message rate and the throughput to the number of members per domain, it can be noticed that this digression happens when the number of domain members increase. The cause of this affect might have a couple of reasons that influenced the results. First, as mentioned earlier, in all case studies, the test setup and the environment parameters has set to the same setting regardless the number of members involved, therefore Data Flow Control (DFC) might affect the message rate and the throughput because number of credits in this module was set statically in a fixed number and was not set according to number of nodes, and when setting the number of credit parameter less than the required, it will slow down the data flow between domain members, this parameter can have a direct effect on both the message rate and the throughput. Second, the network switch might have reached to an overwhelmed state and starts to drop packets which lead to retransmitting the dropped packets again and this process might delay the delivery especially in a reliable packet transmission as in RIDX case.

5.4.2 RIDX Processing Time & Memory Usage

During the experimental tests, a log files were set to collect all the data results from the experimental outputs, the log file includes number of messages sent, the system time in Micro seconds, number of messages sent per second, the throughput per second, the free and total memory in kilobytes. The purpose of this data log file is to calculate the average usage of the memory and the average CPU time needed to finish the transmission task to all the messages. A sample test result for both RIDX UPD as in table 7and Standard TCP as in table 8 for a message size 1K and number of messages (1000000) will be sent by each member simultaneously in a 4 domain members, the throughput is calculated for each member, and the average throughput for all members. The parameters for this test along with the log file interval as follows:

Log Interval	100000 Bytes
Message Size	1000 Bytes
Number of Senders	4 Senders
Number of Messages	1000000 Messages
Number of Members	4 Members

Table 6: The configuration parameters to setup and run the RIDX performance test program

The test results for the configured parameters as follows:

#msgs	SysTime in (ms)	Msgs/Sec	Throughput/Sec [KB]	Free Mem [KB]	TotalMem [KB]
100000	1218786077887	11061.95	11061946.90	330166.07	397869.06
200000	1218786080227	17574.69	17574692.44	347983.49	397869.06
300000	1218786081788	23182.13	23182134.30	399585.12	415629.31
400000	1218786083351	27578.60	27578599.01	250748.66	415367.17
500000	1218786084925	31098.40	31098395.32	281613.07	414973.95
600000	1218786086425	34133.58	34133576.06	288874.70	413663.23
700000	1218786087955	36633.87	36633870.63	384390.09	413728.77
800000	1218786089524	38690.33	38690332.25	245139.47	413859.84
900000	1218786091048	40538.71	40538714.47	339123.56	413794.30

1000000	1218786092598	42103.49	42103490.38	205260.48	413859.84
1100000	1218786094073	43605.80	43605803.54	293514.23	414056.45
1200000	1218786095620	44821.28	44821275.17	387482.82	414253.06
1300000	1218786097080	46045.41	46045407.86	255355.52	414711.81
1400000	1218786098653	46970.41	46970408.64	338720.49	414973.95
1500000	1218786100188	47860.63	47860629.85	183764.79	414842.88
1600000	1218786101681	48729.98	48729975.03	268914.70	415105.02
1700000	1218786103218	49460.30	49460300.84	361359.14	415170.56
1800000	1218786104773	50102.99	50102989.48	223336.59	415694.85
1900000	1218786106270	50770.92	50770916.28	311141.94	415301.63
2000000	1218786107789	51358.43	51358430.49	397084.15	415825.92
2100000	1218786109328	51876.19	51876188.83	286094.67	415694.85
2200000	1218786110869	52353.53	52353529.10	292625.78	414580.74
2300000	1218786112464	52731.73	52731733.04	321395.52	413138.94
2400000	1218786113971	53186.77	53186774.22	348513.61	412483.58
2500000	1218786115437	53659.58	53659583.60	186308.17	412745.73
2600000	1218786117008	53985.59	53985590.00	259691.78	413204.48
2700000	1218786118533	54341.26	54341263.13	309087.55	413597.70
2800000	1218786120079	54653.34	54653341.66	372126.06	413859.84
2900000	1218786121611	54961.72	54961716.32	234858.52	413728.77
3000000	1218786123146	55249.64	55249636.27	311938.29	413401.09
3100000	1218786124614	55588.43	55588430.43	182914.83	413663.23
3200000	1218786126124	55868.85	55868847.88	224531.46	413990.91
3300000	1218786127717	56055.72	56055715.98	272061.95	414253.06
3400000	1218786129437	56114.87	56114870.44	343830.19	414515.20
3500000	1218786131170	56159.04	56159042.41	387312.80	414777.34
3600000	1218786133000	56115.85	56115848.05	223535.50	414646.27
3700000	1218786134732	56158.46	56158457.92	250990.13	414711.81
3800000	1218786136411	56242.97	56242969.63	268208.58	415105.02
3900000	1218786138132	56289.24	56289240.10	281911.22	414580.74
4000000	1218786139941	56263.54	56263538.41	192717.91	415301.63

Table 7: RIDX UDP Test Results

The output results represented in table 7 shows that the average free memory during the transmission period is about 284 Megabyte, and the total memory reserved by RIDX for the full operation is about 403 Megabyte. The actual usage of the total memory averaged to 119 Megabytes. The total physical memory is 4Gegabytes of RAM thus the percentage of the actual memory used is about 3%, and the percentage of the actual memory used from the reserved

memory for this operation is about 30%. These percentages are well acceptable to complete a successful full transmission.

Table 7 represents the log data file for only one RIDX domain member, the other three domain members was setup to log the data in a similar way, and each domain member holds its own log file which contains the same parameters. To consolidate the four domain members' data log, the summary of the four log files are discussed as follows:

- An IP address and a port number was set to identify each domain member, the summation of number of messages received and the time needed to receive all the messages, also the percentage of the loss rate, and the message rate along with the throughput. The summation results per node as the following:

  182.100.3.15:46280:

  Number of messages expected =4000000, Number of messages received=4000000 (loss rate=0.0%), received=4GB, time=71054ms, Messages/Sec=56295.21, throughput=56.3MB

  182.100.0.15:52377:

  Number of messages expected=4000000, Number of messages received=4000000 (loss rate=0.0%), received=4GB, time=71094ms, Messages/Sec=56263.54, throughput=56.26MB

  182.100.2.15:47440:

  Number of messages expected=4000000, Number of messages received=4000000 (loss rate=0.0%), received=4GB, time=71083ms, Messages/Sec=56272.25, throughput=56.27MB

  182.100.1.15:50924:

Number of messages expected=4000000, Number of messages received=4000000 (loss rate=0.0%), received=4GB, time=71094ms, Messages/Sec=56263.54, throughput=56.26MB

- The consolidated data combined from all log files were averaged on both the message rate and the throughput, the results calculated as: (56273.63 Messages/Sec) averaged over all receivers and the throughput is (56.27MB/sec)

- It is noticeable from the results that the loss rate is Zero percent, this means that there was no packets lost during the transmission, and the message transmission using RIDX UDP is a reliable and guaranteed to deliver all the messages, and retransmit any undelivered message.

5.4.3 TCP Processing Time & Memory Usage

The case study conducted in section 5.3.1 was done for both RIDX UDP and standard TCP. The data test results using TCP protocol was collected in data log files, the log file parameters includes number of messages sent, the system time in Micro seconds, number of messages sent per second, the throughput per second, the free and total memory in kilobytes. Table (8) represents the data test results for a message size 1K and the number of messages (1000000) will be sent by each member simultaneously in a 4 domain members, the throughput is calculated for each member and the average throughput for all members. The TCP transport protocol data log results is as follows:

#msgs	SysTime in (ms)	Msgs/Sec	Throughput/Sec [KB]	Free Mem [KB]	TotalMem [KB]
100000	1218786691748	9545.63	9545628.1	218965.75	397869.06
200000	1218786693883	15859.17	15859170.57	369874.02	397869.06
300000	1218786695716	20769.87	20769869.84	237457.40	414973.95
400000	1218786697553	24568.52	24568515.45	311860.92	412352.51
500000	1218786699419	27552.76	27552763.54	354295.24	412745.73

600000	1218786701239	30049.58	30049581.81	193283.33	412745.73
700000	1218786703028	32175.03	32175032.18	208764.91	413007.87
800000	1218786704857	33919.86	33919864.32	276655.47	413007.87
900000	1218786706574	35570.31	35570310.65	232425.24	413007.87
1000000	1218786708346	36935.81	36935805.57	184918.25	413401.09
1100000	1218786710139	38105.80	38105795.55	221633.88	413138.94
1200000	1218786711948	39118.53	39118529.14	243869.42	413466.62
1300000	1218786713725	40057.93	40057929.93	271953.07	413794.30
1400000	1218786715496	40906.97	40906965.87	379450.01	413794.30
1500000	1218786717245	41697.94	41697940.12	252284.27	413204.48
1600000	1218786719041	42362.78	42362784.29	335518.22	412483.58
1700000	1218786720851	42952.07	42952070.54	188578.70	411303.94
1800000	1218786722625	43527.68	43527676.35	265635.89	410845.18
1900000	1218786724433	44021.22	44021222.86	334603.47	410386.43
2000000	1218786726248	44468.16	44468160.80	383264.92	411041.79
2100000	1218786727969	44970.77	44970769.00	362625.70	404881.41
2200000	1218786729743	45387.96	45387963.94	184358.97	411041.79
2300000	1218786731499	45792.10	45792103.85	300320.62	411041.79
2400000	1218786733276	46150.30	46150296.13	324522.50	411041.79
2500000	1218786735067	46472.72	46472720.51	267699.86	409468.93
2600000	1218786736819	46807.21	46807208.31	281638.55	410845.18
2700000	1218786738633	47070.31	47070309.10	307599.27	411369.47
2800000	1218786740446	47318.08	47318078.89	330946.09	411762.69
2900000	1218786742221	47580.76	47580764.25	345182.05	412155.90
3000000	1218786743966	47851.47	47851469.04	368708.06	412680.19
3100000	1218786745748	48079.91	48079905.70	379184.38	413138.94
3200000	1218786747528	48297.51	48297512.68	244953.00	411697.15
3300000	1218786749272	48529.41	48529411.76	286501.25	412024.83
3400000	1218786751003	48758.80	48758801.68	199740.78	412614.66
3500000	1218786752774	48949.68	48949679.73	332359.59	412680.19
3600000	1218786754851	48927.00	48927003.63	247119.20	412614.66
3700000	1218786756644	49089.85	49089847.69	380152.54	412942.34
3800000	1218786758533	49183.93	49183934.97	298092.46	413663.23
3900000	1218786760390	49293.46	49293460.40	215528.99	413466.62
4000000	1218786762344	49338.86	49338859.29	351471.90	413925.38

Table 8: Standard TCP Test Results

The output results represented in table 8 shows that the average free memory during the transmission period is about 280 Megabyte, and the total memory reserved by RIDX for the full operation is about 402 Megabyte. The actual usage of the total memory averaged approximately 122 Megabytes. The total physical memory is 4Gegabytes of RAM thus the percentage of the actual memory used is about 3%, and the percentage of the actual memory used from the reserved memory for this operation is approximately 31%.

Every domain member was set to log the data individually, and each domain member contains the same log file parameters. The consolidation of the data for the four domain members was summed up as the following:

- Using TCP protocol, a combination of IP address and a port number identifies each domain member, considered that the system is installed on separate units. A common port number opened and known to all domain members to establish the connection. The connection initiated based on a 3 way handshaking technique and terminates when all the message transmission is accomplished. The summation is calculated to include number of messages received and the time needed to receive all the messages, also the percentage of the loss rate, and the message rate along with the throughput. The summation results per node as the following:

  182.100.3.15:7700:

  Number of messages expected =4000000, Number of messages received=4000000 (loss rate=0.0%), received=4GB, time= 81073ms, Messages/Sec= 49338.25, throughput=49.34MB

  182.100.0.15:7700:

Number of messages expected=4000000, Number of messages received=4000000 (loss rate=0.0%), received=4GB, time= 81072ms, Messages/Sec= 49338.86, throughput=49.34MB

182.100.2.15:7700:

Number of messages expected=4000000, Number of messages received=4000000 (loss rate=0.0%), received=4GB, time=81044ms, Messages/Sec=49355.91, throughput=49.63MB

182.100.1.15:7700:

Number of messages expected=4000000, Number of messages received=4000000 (loss rate=0.0%), received=4GB, time=81037ms, Messages/Sec=49360.17, throughput=49.36MB

- The averaged data for message rate and the throughput summed as: (49348.30) Messages/Sec averaged over all receivers and the throughput is (49.35MB/sec)
- In standard TCP transmission protocol the delivery of packets is a Unicast (point-to-point) reliable delivery, therefore It shows in the results that the loss rate is also (0.0) percent.
- When comparing CPU time and memory usage in RIDX and TCP, RIDX has a slight improvement over TCP in terms of utilizing the memory usage and the CPU time needed to accomplish the transmission.

5.5 SUMMARY OF PERFORMANCE TEST RETULTS

In the performance test results, the throughput and the message rate is trivially demeaned when we increase the number of members in the domain. But it is noticeable from all case studies that in most cases RIDX has achieved better message rate and better throughput, especially on a 1K message size, the improvement on 1K message size has a significant

advancement regardless the number of domain members, furthermore, the utilization of memory and CPU processing time using 1K message size (less than MTU size) has also a significant improvement. On the other hand, when using 2.5K and 5K message sizes, the results showed that RIDX has achieved a slight improvement over TCP, and in other cases, the results averaged nearly to each other, the probable cause is that the Data Flow Control has not many credits to maintain a continuous data flow without any delays or the network switch begins dropping data packets. The current DFC module does not have a dynamic setting for the bank credit to adjust the pre-setting credit numbers according to number of domain members. The throughput and the message rate for all messages can be collectively averaged over the whole domain, and then it can be scaled the rate up almost around 100%, that means we can distribute 5,000,000 messages of 1K in a domain with 10 members, and the throughput can reach over 800MB/Sec.

Another factor that can positively affect the performance of RIDX is the size of the Maximum Transmission Unit (MTU), in all previous results, the MTU size was set to its default size (1500 Bytes). A preliminary test results showed even better performance when setting the MTU to 9000 bytes (Jumbo Frames). Using the same hardware environment setup as shown in case studies, but setting the MTU to a Jumbo Frame size (9000 bytes), the results showed that the achievement in a domain of 4 members and each member sends 5,000,000 message of a 1K size has jumped from 56MB/Sec to 137MB/Sec. On a 2.5K message size, the throughput jumped from 94MB/Sec to 149MB/sec. On a 5K message size, the throughput has raised to 155MB/Sec.

5.6 A COMPARISON CASE STUDY

Eric Gamess and Rina Suros (2008) have made a performance test and measure the throughput for both IPv4 and IPv6, and made a comparison of the throughput between the two

TCP versions. They tested a point-to-point connection between TCP/IP and UDP transport protocols for transferring the data. In addition, they did a test comparison using different platforms (e.g. Windows XP 2, Solaris 10 and Debian 3.1). They did the same performance test to measure the throughput between IPv4 and IPv6 UDP protocol on the same environment. The MTU has a length of 1500 bytes, and the RAM has 2GB with a dual core AMD CPU (2GHz). They calculated the maximum (theoretical) throughput as shown in the figure below for a point-to-point connection:

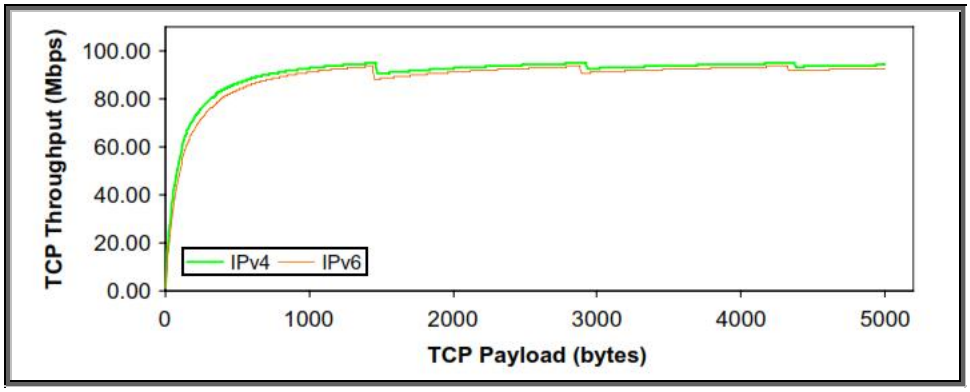

Figure 25: Max Theoretical Throughput (Eric Gamess and Rina Suros 2008)

The data payload in both IPv4 & IPv6 varies between 1 byte to 5K (5000 bytes), and as noticed in the graph above, the lower curve is for the IPv6 TCP version, this made the superior upper one (IPv4) due to the size of the IP message header, for IPv6 is 40 bytes as for the IPv4 is only 20 bytes, and effected the theoretical throughput of about 5% difference.

The actual test results to measure the throughput has been made on different platforms (Windows, Solaris and Debian) against the Maximum theoretical throughput, the next graph represents the throughput for IPv4 TCP between different platforms (operating systems):

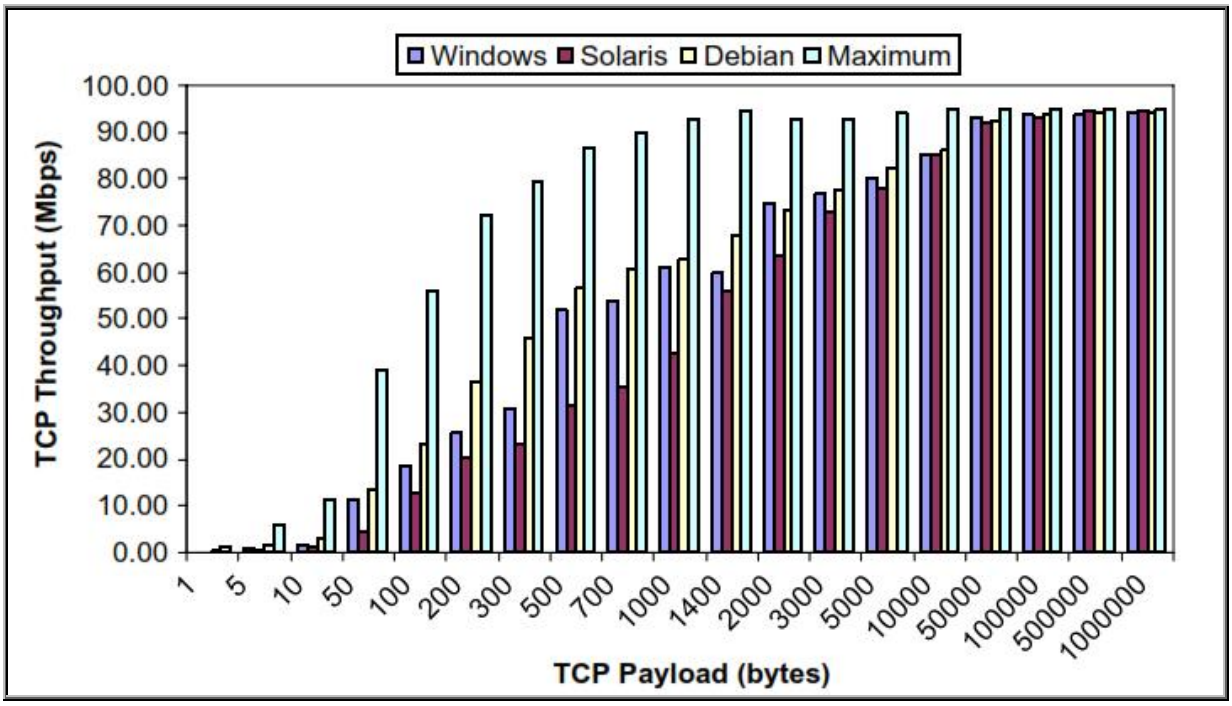

Figure 26: IPv4 Comparison Throughput (Eric Gamess and Rina Suros 2008)

In their findings, they estimated the difference in the performance between IPv4 and IPv6 is about 5%. As noticed from the outputs, there is an amount of variation between different platforms. In figure (26) showed that the performance for example Windows platform achieved better performance in certain message sizes, but in other cases Debian platform gave better performance than others. Therefore, the performance between the three platforms under the same environment test varied depending on message size, for that reason, when comparing these findings for these different platforms used, we need to calculate the average throughput value on each message size in different platforms, and then compare the mean values with the RIDX UDP throughput. It needs to be mentioned here some facts for better understanding of the comparison:

1- The four RIDX domain members model is chosen for comparison because it represents the simplest module to be compared with against point-to-point communication module.

2- The four RIDX domain members model contains the main factors to complete the benchmark of RIDX architecture, and the ideal model to represent a multi path virtual domain network that contains a sender, a receiver and two routers, and has an acceptable performance outcome in terms of throughput and message rate.

3- This case study uses a point to point transmission, which are one source and one destination only.

4- A Unicast UDP transport protocol measurement from this case study is included in this comparison; therefore, this can be another performance indicator between Unicast UDP transmission method against Multicast RIDX UDP transmission method.

Comparison Throughput results

	1K	2.5K	5K
RIDX UDP	56.27	94.24	107.92
TCP Ipv4	52.01	80.21	85.33
TCP Ipv6	51.2	78.76	82.44
UDP Ipv4	52.87	82.43	86.53
UDP Ipv6	52.12	80.33	85.04

Table 9: Comparison throughput results

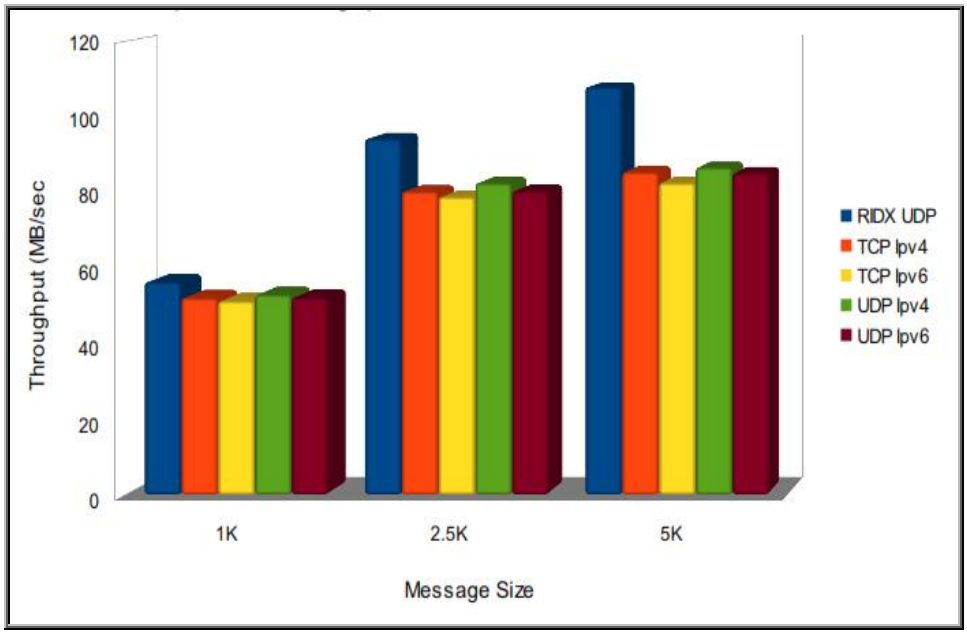

Figure 27: A comparison throughput between RIDX & other case studies

We notice from the comparison table results in Table 9 and the Bar graph figure (27) that RIDX has a significant better performance than the standard TCP transport protocol, at the same time, preserving the security measurements that RIDX architecture is enforcing. As a result, we can conclude that reliable multicast transmission showed better performance than a single point to point transmission.

CHAPTER 6

6. Conclusions & Recommendations

6.1 CONCLUSIONS

Security data threats are always subject under improvement & development. Securing XML file during transmission is one of the challenges system developers can face, protecting XML data from any threats, attacks and exposure during transmission is an essential subject to be thoroughly studied. Different techniques are applied to protect XML data, but this study proposed architecture of how to build an embedded multilevel line of defence for securing XML data during transmission between different organizations, businesses and services. The experimental test is made to measure the performance, considering preserving the security methods and techniques that was implemented in this approach. The experimental results showed that RIDX UDP transport protocol has better performance than a standard TCP transport protocol. As a result, protecting the data and maintaining the transmission performance coupled with a reliable delivery is an achievement to this research.

As a summary, achievements summed up as follows:

- Multi agent domain members spread across the internet cloud WAN and LAN, with ability to form a secured virtual domain network that can initialises, joins, leaves the domain in an automated process flow.

- An automated authentication process during member joining stage, and automated Public/Private encryption key mechanism for all messages during transmission between domain members
- Automatic detection for all nodes that joins, leaves, and crashes, then notifies all domain members and update the Active Member List for the use of routing and other services.
- Vertical XML file shredding into several chunks, each chunk contains a combination of unrelated set of characters to hide XML file structure and conceal confidential information.
- Use RIDX UDP transport protocol to transmit XML data in a reliable and multicast transmission as a bottom-most transfer protocol in order to enhance transmission efficiency in terms of message rate, throughput, and memory/CPU usage.
- Routing capabilities to route messages between multi agent domain group members
- An automated Tunnelling technique process between members to add another security layer to the communication pipelines.

The conclusion summary is that when creating a virtual multi agent domain group network using RIDX UDP protocol prior to data transmission improve the communication efficiency and enhances the transmission by achieving higher message rate and throughput. Furthermore, embedding and automating security measurements in RIDX such as Authentication, Partitioning, Encryption and Tunnelling has lowered the operational running cost, lowered the complexity of achieving the security layers, and utilized the resources usage. Adding to this lightweight infrastructure to satisfy high security measurements and

simultaneously provide a global framework for interoperability by an efficient bottom-most transport layer for data exchange in an unsecured environment.

6.2 RECOMMENDATIONS FOR FUTURE WORK

RIDX is a middleware for transmitting XML data files that uses the internet cloud as a platform. Many factors can affect this approach due to wide variances that can exist in the internet cloud, such as using different hardware, different operating systems, different routing techniques and different security measures. Some improvements can be considered to this approach to expand the capabilities of the system, the following is the summary of these concerns and future work:

- A comparative study and a thorough testing of data vulnerability for the embedded security methods used in RIDX and the formal security methods
- Test, Analyse and study the communication efficiency for message rate and throughput when scaling up the number of domain members over 50 or 100 groups members together in one single domain group.
- Studying and Analysing in depth the effect of various security attacks on RIDX system before and during data transmission.
- When registering to a domain, all members in the domain works as a group, regardless the geographical locations, a case my arose when the ISP (Internet Service Providers) or any middle routing hardware might fail or crashes, which results to separating the domain members into more than one group, which will lead to electing domain coordinator for each separated group, therefore, a mechanism should be applied to

rejoin and merge back all groups into one domain group whenever the routing connection has been restored.

- Due to the factor of using different hardware and routing techniques, packet speed varies depending on the route the message has taken, creating a slow delivery for some packets to the final destination. Therefore, a module can solve this existing downside by calculating network fastest and least cost path between all RIDX members. A list of members with best routing performance options can be distributed to be used as preferred routing list members.

- Tunnelling experiments showed a significant overhead processing in both throughput and the packet delays. A study could be made to compare the throughput and the latency between the software tunnelling implementation and the hardware implementation, or use different types of transport protocols and compare the throughput and the latency before and after tunnelling.

- Study the effect of compression techniques on the system performance in terms of message rate and throughput and the usage of the resources.

- Study the network efficiency when using RIDX as middleware for Wireless and mobile implementations

- Enhancement can be made to the Data Flow Control module by adapting a dynamic credit bank technique, instead of pre-configuring number of credits; the new enhancement could be done by calculating the free memory allocations and packet loss rate dynamically.

References

Abiteboul, S. (1993) Querying and updating the file. In Proceedings of 19th International Conference on Very Large Databases, Dublin, Ireland, p. 73 -- 84

Abiteboul, S. (1997) Queries and Computation on the Web. International Conference on Database Theory ICDT, p. 262 -- 275

Abiteboul, S. (1997) Querying Semi Structured Data. In Proceedings of International Conference on Database Theory ICDT, p. 1 -- 18

Abiteboul, S. (1997). The Lorel Query Language for Semistrucutred data. International Journal on Digital Libraries, Volume 1, Issue 1, p. 68 -- 88

Abiteboul, S. (1999) Data on the Web: From Relations to Semi structured Data and XML, Morgan Kaufmann, ISBN:1-55860-622-668

Abiteboul, S. (2001) Semistructured Data: from Practice to Theory. In Proceedings of the 16th Annual IEEE Symposium on Logic in Computer Science LICS, IEEE Computer Society, Washington, DC, United States of America, p. 379

Adeel, M. and Iqbal A. (2004). "TCP Congestion Window Optimization for CDMA2000 Packet Data Networks" International Conference on Information Technology (ITNG'07): 31–35. doi:10.1109/ITNG.2007.190. ISBN 978-0-7695-2776-5

Al-Wasil, F. (2006) Query Translation for Distributed Heterogeneous Structured and Semi-structured Databases. British National Conference on Databases BNCOD, Belfast, p. 73 -- 85

Al-Wasil, F. (2006a). Establishing an XML metadata knowledge base to assist integration of structured and semi-structured databases. In Proceedings of the 17th Australasian Database Conference. Volume 49. Hobart, Australia, p. 69 -- 78.

Amer-Yahia, S. (2002) A Mapping Schema and Interface for XML Stores. In Proceedings of the 4th international Workshop on Web information and Data Management WIDM, Mclean, Virginia, United State of America.

Ardagna, C., Damiani E., , Vimercati, S. and Samarati, P. (2007) "Security, Privacy, and Trust in Modern Data Management Data-Centric Systems and Applications", Part II, 71-86, DOI: 10.1007/978-3-540-69861-6_6

Arnab, A., & Hutchion, A. (2005). “Requirement analysis of enterprise DRM systems”

Arnold, J. (2010)"Why businesses need to think differently about cloud communications"

Atay, M. (2004), “Mapping XML data to relational data: A DOM-based approach”. In 8th IASTED International Conference on Internet and Multimedia Systems and Applications, Kauai, Hawaii, p. 59 -- 64

Atay, M. (2007), “Efficient schema based XML-to-Relational data mapping”. Information Systems, Volume 32 (3), p. 458 -- 476

Atwood, J. and Barhoush, M..(2010) Telecommunication Systems Vol: 45 Issue: 1 ISSN: 1018-4864 Page 3-20: Digital rights management (DRM).

Balmin, A (2005) Storing and Querying XML data using demoralized relational databases The VLDB Journal, Volume 14, p. 30 -- 49

Barbosa, D. (2001). ToX: The Toronto XML Engine. In Proceedings of the Workshop on Information Integration on the Web, Rio de Janeiro, Argentina

Barre, P., Paasch, C. And Bonaventure, O. (2011). "MultiPath TCP: From Theory to Practice" IFIP Networking

Barre, R., Greenhalgh, P. and Handley, W. (2011). "Improving datacenter performance and robustness with multipath TCP", SIGCOMM '11 Proceedings of the ACM SIGCOMM 2011 conference, ISBN: 978-1-4503-0797-0

Biddle, P., England, P., Peinado, M., and Willman, B. The darknet and the future of content distribution. http://msl1.mit.edu/ESD10/docs/darknet5.pdf . Accessed [20.12.2008].

Birman, K., Hillman, R. and Pleisch, S. (2005). "Building network-centri military applications over service oriented architecture". "SPIE Defence and Security Symposium" Orlando - Florida

Blahut, R. (2004)"Algebraic Codes for Data Transmission" Cambridge University Press, ISBN 0521553741

Bryant, D., Atallah, J. and Stytz, R. (2004). "A survey of Anti-Tamper technologies CrossTalk". The Journal of Defense Software Engineering, 17(11), 12–16. http://www.arxan.com/ATknowledgePortal/pdfs/Crosstalk-Article-A-Surve-of-AntiTamper-Technologies.pdf . [Accessed 11.11.2008].

Buneman, P. (1996) "A query language and optimization techniques for unstructured data" In Proceedings of ACMSIGMOD International Conference on Management of Data, Montreal, Canada, p. 505-- 516

Buneman, P. (1997) "Adding Structure to unstructured data", In proceeding of the international conference on Database Theory, Springer Verlag, p. 336 -- 350

Buneman, P. (1997) "Semistructured Data" Symposium On Principles Of

Buneman, P. (2000) UnQL: “A query language and algebra for semi structured data based on structural recursion”. VLDB Journal.

Buneman, P., Choi, B., Fan, W., Mann, R., Hutchison, R. And Viglas, S. (2005) Vectorizing and Querying Large XML Repositories.

Bushell-Embling, D. (2010) "Malaysia's pricey leased lines", Telecom Asia Vol: 21 Issue: 6 ISSN: 1681-181X.

Bustamante, F., Eisenhauer, G., Schwan, K., and Widener, P, (2000), “Efficient wire formats for high performance computing”, In Proceedings of the ACM/IEEE 2000 Conference on Supercomputing.

Cattell, R. (2000) “The Object Data Standard ODMG 3.0. Morgan Kaufmann”, ISBN: 1558606475.

Chamberlin, D. (2000) “An XML Query Language for Heterogeneous Data Sources”. International Workshop on the Web and Databases (WebDB'2000), Dallas, United State of America.

Chaudhuri, S. (2005) “Storing XML (with XSD) in SQL Databases: Interplay of Logical and Physical Designs”. IEEE transactions on knowledge and data engineering, volume 17 (12).

Chawathe, S. (1994). The TSIMMIS Project: “Integration of Heterogeneous Information Sources”. In Proceedings of 10th Meeting of the Information Processing Society Conference, Tokyo, Japan, p. 7 -- 18

Chen, J., Hu C., and Ji,Z. (2011) Self-Tuning Random Early Detection Algorithm to Improve Performance of Network Transmission, Educational Researcher Vol: 39 Issue: 7 ISSN: 0013-189X ,Pages: 515 – 524,

Christophides, V. (1994) from structured documents to novel query facilities. SIGMOD Record, Volume 23 (2), p. 313 – 324

Database Systems PODS, p. 117 – 121

Database Systems TODS, Volume 4, Issue 4, p. 531 – 544

Davis, D. and Parashar, M. (2002) "Latency performance of SOAP implementations". In Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, pages 407-412.

Deering S. and Hinden R. (1998). "Internet Protocol, Version 6 (IPv6) Specification, Internet Engineering Task Force" RFC 2460

Douglas C. (1995) "Internetworking with TCP/IP, Volume 1: Principles, Protocols, and Architecture", Prentice Hall, ISBN 0132169878.

Douglas C. (2006), "Internetworking with TCP/IP: Principles, Protocols, and Architecture". (5th Ed.). Prentice Hall, ISBN 0131876716.

Duftler, C., Khalaf, M.,Nagy R., Mukhi W. and Weerawarana, S. (2002): Unraveling the web services web: An introduction to SOAP, WSDL, UDDI. IEEE Internet Computing, 6(2): 86-93.

Emina, S., Liu, R., and Spasojevic, P. (2004) "Hybrid ARQ with Random Transmission Assignments", http://ect.bell-labs.com/who/emina/papers/hatc.pdf

Fan, W., Chan, C. and Garofalakis, M. (2004) "Secure XML querying with security views", SIGMOD '04 Proceedings of the 2004 ACM SIGMOD international conference on Management of data. ISBN:1-58113-859-8

Felten, W., and Halderman,, A. (2006) “digital rights management, spyware, and security”. Los Alamitos: IEEE Comput.Soc.

Fernando G. (2009). "On the implementation of TCP urgent data", 73rd IETF meeting Gregorio J. URI Templates. http://bitworking.org/projects/ URI-Templates/

Fielding, R. (2007) “A little REST and Relaxation” The International Conference on Java Technology (JAZOON07), Zurich, Switzerland

FIX Protocol Ltd. *FIXML:* A markup language for the FIX application message layer. Version 4.4. http://www.fixprotocol.org/specifications/fix4.4fixml , [accessed 1.9.2010].

FIX Protocol Ltd. The Financial Information Exchange Protocol (FIX), version 5.0, http://www.fixprotocol.org/specifications/FIX.5.0SP2 , [accessed 11.11.2009].

FIXML – FpML Overview of the schema FIXML 4.4, http://www.fixprotocol.org/specifications/fix4.4fixml , [accessed 10.9.2010]

Gamess, E. and, Suro, R. (2011) "An upper bound model for TCP and UDP throughput in IPv4 and IPv6", Internet Technology and Applications (iTAP), 2011 International Conference. ISBN: 978-1-4244-7253-6, INSPEC Accession Number: 12215982

Harms, J. and Holte, R. (2005) "Impact of Lossy Links on Performance of Multihop Wireless Networks", IEEE, Proceedings of the 14th International Conference on Computer Communications and Networks: 303 - 308

http://www.gont.com.ar/talks/hacklu2009/fgont-hacklu2009-tcp-security.pdf TCP DoS (Denial of Service) vulnerabilities [2009].

http://www.ipv6tf.org/index.php?id=3882&lan=en&page=news/newsroom Action Plan for the deployment of Internet Protocol version 6 (IPv6) in Europe the IPv6 Portal, published 2008, [accessed 18.6.2011].

In Proceedings information security South Africa, Hotel Balalaika, Sandton, Johannesburg. http://pubs.cs.uct.ac.za/archive/00000205/01 [Accessed 15.12.2008].

Ioannidis, S. Keromytis, D., Bellovin,.M. and Smith M., (2000) "Implementing a Distributed Firewall", Proceedings of Computer and Communications Security (CCS), pp. 190–199

Jacobi, J. (2011) “Optimize Your Router for VoIP and Video Streams “, PC World Vol: 29 Issue: 3 ISSN: 0737-8939, Start Page: 89,

Kenneth P. Chen, B, Hopkinson, K., Thomas, R.,Thorp, J., Renesse, R., and Vogels W. (2005) “Overcoming Communications Challenges in Software for Monitoring and Controlling Power Systems” IEEE. Vol. 9, No. 5. International Conference on Data Engineering, ICDE, Tokyo, Japan.

Kent, C. and ,Mogul.J. (1995) "Fragmentation Considered Harmful" , ACM SIGCOMM Computer Communication Review - Special twenty-fifth anniversary issue. Highlights from 25 years of the Computer Communication Volume 25

Kim, S. (2002) “A Data Model and Algebra for Document-Centric XML Document” Information Networking, Wireless Communications Technologies and Network Applications, International Conference. Cheju Island, Korea, Volume 2, p. 714 – 723

Kohloff, C. and Steele, R. (2003) “Evaluating SOAP for High Performance Business

Applications: Real-Time Trading Systems, Systems, http://www2003.org/cdrom/papers/alternate/P872/p872\kohl

Kudrass, T. (2002) “Management of XML Documents in Object- Relational Databases” EDBT 2002 workshops, Lecture Notes in Computer Science 2490, Springer-Verlag, p. 210—227

Kurose, F., and Ross, W. (2010) “Computer Networking: A Top-Down Approach (5th ed.)”. Bosto, MA: Pearson Education. ISBN 9780131365483

Lau, H., Ng, W., Yang, Y. and Cheng J. (2006) "An efficient Approach to Support Querying Secure Outsourced XML Information", CAiSE'06 Proceedings of the 18th international conference on Advanced Information Systems Engineering, Springer-Verlag Berlin, Heidelberg. ISBN:3-540-34652-X 978-3-540-34652-4

Leonov, A. (2004) Construction of an Optimal Relational Schema for Storing XML Documents in an RDBMS without Using DTD/XML Schema Programming and Computer Software, Volume 30 (6), p. 323—336

Martin, G. Dimitris, A. and Roland, W. (1998) "Cost-efficient network synthesis from leased lines",Annals of Operations Research Vol: 76 Issue: 0 ISSN: 0254-5330, Pages: 1-20.

Mazurczyk, W, and Szczypiorski, K., (2009) “Cryptography and Security” . http://arxiv.org/abs/1006.0495

McHugh, J. and Widom, J. (1999) Query Optimization for XML. In Proceedings 25th International Conference on Very Large Databases, Morgan Kaufmann, p. 315—326

Miklau, G. (2006) "The XML data repository", http://www.cs.washington.edu /research/xmldatasets/

Möller, B., and Löf, S. (2009) “A Management Overview of the HLA Evolved Web Service API”. http://www.pitch.se/images/06f-siw-024.pdf

Murphy, D. and Heffernan, D. (2003) “Assembly Automation” Vol: 23 Issue: 1 ISSN: 0144-5154 Pages: 60 - 68.

Novak, L. and Zamulin, A. (2005) “Algebraic Semantics of XML Schema”. Proceding ADBIS 2005, LNCS, Volume 3631, p. 209 – 222

Olovsson, T. and John, W. (2008) “Detection of malicious traffic on back-bone links via packet header analysis”, Campus-Wide Information Systems Vol: 25 Issue: 5 ISSN: 1065-0741 Pages: 342 - 358

Pautasso, C., Zimmermann, O., and Leymann, F. (2008) "RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision", 17th International World Wide Web Conference (Beijing, China)

Perez, A., Zarate, V., Montes, A. and Garcia, C. (2006) "Quality of Service Analysis of IPSec VPNs for Voice and Video Traffic," Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/Advanced International Conference on

Peterson, L. and Davie, B. (2000) "Computer Networks: A Systems Approach", Morgan Kaufmann, ISBN 15586051428888

Popoviciu C, Levy-Abegnoli E. and Grossetete P. (2006) "Deploying IPv6 networks", 1st ed. Cisco Press

Rabhi, F. and Benatallah, B. (2002) “An integrated service architecture for managing capital market systems” IEEE Network, 16(1)

RFC 2460 (1998) Internet Protocol, Version 6 (IPv6) Specification, S. Deering, R. Hinden

Rittinghouse, J. and Ransome, J. (2009) "Cloud Computing: Implementation, Management, and Security", Edition: 1 Call Num: 004.36 ISBN: 1439806802, Pages: 340.

Rizzolo, F., Mendelzon A. (2001) “Indexing XML Data with ToXin”. WebDB p. 49 - 54

Robie, J. (1998) “XML Query Language (XQL) online” Available from http://www.w3.org/TandS/QL/QL98/pp/xql.html [Accessed 13.122008]

Rosasco, N. and Larochelle, D. (2003) “How and Why More Secure Technologies Succeed in Legacy Markets: Lessons from the Success of SSH", Dept. of Computer Science, Univ. of Virginia. http://www.cs.virginia.edu/~drl7x/sshVsTelnetWeb3.pdf

Schaffer, D., (2011) "Network security in the Automation world ", InTech Vol: 58 Issue: 2 ISSN: 0192-303X,P. 58.

Serrão, C., Dias, M., and Delgado, J. (2006) Bringing DRM Interoperability to Digital Content Rendering Applications, 323-330, DOI: 10.1007/1-4020-5261-8_50.

Shiau, W., Li, Y., Chao, H. and Hsu P. (2006) "Evaluating IPv6 on a large-scale network Computer Communications archive Volume 29 Issue 16, Amsterdam, The Netherlands. 29(16):3113–21.

Staken, K. (2001) “Introduction to Native XML Database [online]”. Available from: http://www.xml.com/lpt/a/2001/10/31/nativexmldb.html [Accessed 29.11.2007]

Stanton, R., (2005) "Securing VPNs: comparing SSL and IPsec", Computer Fraud & Security Vol: 2005 Issue: 9 ISSN: 1361-3723, Pages: 17-19

Thornycroft, P. (2010) "Moving communications into the cloud", http://agentsandbrokers.phoneplusmag.com/articles/moving-communications-into-the-cloud.html

Weinberg J (2010) “2G Applications, An emerging media channel in developing nations”. http://nydigitallab.ogilvy.com/2010/05/19/2g-applications-an-emerging-media-channel-in-developing-nations/ [accessed 25.3.2011]

Whitemore, J. (2009) "Unified communications for the enterprise: hosted VoIP or cloud communications", Smoothstone IP Communications. http://www.westipc.com/blog/2009/06/04/hosted-voip-or-cloud-communications/

Wilkinson, P. (2005) ” Construction Collaboration Technologies: The Extranet Evolution”, ISBN 0-415-35859-0

Yu, J. and Liu C., (2006) "Performance Analysis of Mobile VPN Architecture," 4th Annual Conference on Telecommunications, and Information Technology, Las Vegas

Zuleita H., Lau, V. and Cheng, R., (2009) "Cross-Layer Design of FDD-OFDM Systems based on ACK/NAK Feedbacks". Information Theory, IEEE Transactions on Volume: 55, Issue: 10 On Page(s): 4568 - 4584

zur-Muehlen, M.; Nickerson, J. and Swenson, K,, (2005) “Developing Web Services Choreography Standards” – The Case of REST vs. SOAP. Decision Support Systems 37, Elsevier, North Holland, Volume 40 Issue 1