A MIDDLEWARE FOR DISTRIBUTING XML DATA BETWEEN MOBILE APPLICATION SERVERS

Yousef. E. Rabadi and Joan. Lu

University Of Huddersfield

Author Note

Yousef E. Rabadi and Joan Lu, School of Computing and Engineering, University of Huddersfield

Correspondence concerning this article should be addressed to Yousef E. Rabadi, School of computing & Engineering, University of Huddersfield, Queensgate, Huddersfield HD1 3DH, UK. Email: Yousef.rabadi@hud.ac.uk

ABSTRACT

This research seeks to introduce architecture of new approach of distributing XML data files between different mobile application servers. The importance of this study is to set a multi level of security defense for interchanging XML data files between different servers. The main objective and goal of this study is to transmit XML data files between different Mobile Application Server (MAS) using internet cloud infrastructure in a secured manner coupled with reliability and quality of communication. Taking into consideration that the system architecture attribute is to be independent, scalable and flexible of using cloud computing. Furthermore, this architecture designed to minimize the risk of any alteration, data loss, data abuse, data misuse of XML critical business data information. As cloud computing, using existing cloud network infrastructure to get advantage of the scalability, operational efficiency, and control of data flow are big consideration in this architecture. A test has been made to measure the performance of the Real-time Interactive Data Exchange system (RIDX), one by using standard TCP protocol, and one by using RIDX UDP protocol. As a result, starting from 4 nodes up to 10 nodes in the cloud, RIDX architecture performance showed good results, conversely the study showed that using RIDX UDP protocol as a transport protocol gives better performance than standard TCP, moreover, using RIDX UDP transport protocol assures the reliability and lossless of data transmission to all nodes, therefore, RIDX acts as a reliable multicast transmission.

Keywords XML Security, Mobile application server, Cloud Computing, XML distribution, (TCP, UDP, Multicast) Protocols

A Middleware for Distributing XML Data between Mobile Application Servers

Exchanging data between different domains plays an important and essential role of doing businesses. As mobile applications use the internet as source of information, and these information is processed behind the scene by mobile application servers, some of the application servers adapted XML file structure to be as the transmission data format, therefore, XML increasingly become a standard format for transmitting data on networks between different businesses, the need of finding secured and efficient techniques of transmitting XML data files is a necessity matter.

Real-Time XML Data transmission aims to place a multi layer defense of exchanging XML data information electronically between different mobile application servers using internet cloud network as a platform. RIDX (Real-time interactive data exchange) 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 being transferred. The proposed system adapts as part of its transport protocol by using multicast data distribution mechanism in order to reduce network resources, minimize the hindrance and expenditure.

Thus far, there has been little stimulus to use multicast data distribution, because it lacks of protection mechanism for the data been delivered.

Although there has been a significant progress in presenting secured multicast data distribution.

Different drawbacks, threats and attacks can be addressed during single/bidirectional transmission process:

- Communication channels that are not secured will jeopardize 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 of providing its ordinary functions which results of non-contemporary information, this Denial of service (DoS) can cause the organization a great pact of service time and money
- When transmitting data across shared network, some tools can be used to sniff network packets in order to reveal sensitive information especially if no data encryption measurements has been taken

By taking these threats into consideration of building a system, the belief that this new approach contains the 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 stage of defense is the key to this project:

- Level 1: Shredding and Encrypting/decrypting the data
- Level 2: Securing the communication (tunneling)
- Level 3: Reliable Multicast delivery, routing and merging the data

METHOD

Related Work

One of the protocols and services has been used to transmit XML data format is SOAP (Simple Object Access Protocol). Evaluation studies of SOAP XML transmission protocol has been made, specifically of the SOAP and XML performance, several of them concluded that the performance of SOAP and XML acquire considerable price when compared to binary transmission protocols.
An evaluation experiment was conducted about SOAP implementation latency performance, comparing with CORBA/IIOP and Java RMI protocols. As a

conclusion, SOAP is enormously slower, therefore, SOAP XML messages is not appropriate for transmitting bulk data.

Moreover, A comparison has been made between SOAP and binary CDR and FIX protocol, the study showed that SOAP did fare poorly compared with both binary CDR and FIX protocol.

Another implementation of exchanging messages is the Financial Information eXchange (FIX) protocol, which is a messaging standard developed specifically for the real-time electronic exchange of securities transactions and stock markets. FIX messages consist of tag-value pairs separated by a special delimiter character (SOH, which is ASCII value 0x01), and types of values include strings, integers, floating point values, it is fast and reliable protocol 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, it is used also for pure financial interchange trading system, and does not solve the XML data file transmission between different applications or different types of businesses, furthermore, 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 which is REST (Representational State Transfer), it is a Web Services depend on method of Request-Respond over HTTP protocol, methods such as (PUT, GET, POST…etc), 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”.

Contribution & Importance of this Study:

A major contribution of this study is to ensure service availability and to sustain service on demand by means of using the internet cloud as a platform, and to conduct a secured XML data exchange between domain groups, businesses and organizations. Keeping in mind the internet security threats, therefore build multi level security prevention and defense mechanism in order to transmit XML data file in a secured manner coupled with reliability, quality of communication, independent, scalable, and flexibility.

Another major contribution for this system architecture is to implement multicast reliable communication to deliver XML data file in a multi concurrent connections, with no effect or delay to other concurrent transmissions, this will widen the simultaneous number of connections to serve data transmissions. Using a multicast reliable communication technique can get advantage of the availability of RIDX instances which resides in the internet cloud; this will allow the sender to elect the most efficient nodes including the utilized communication paths and routers that will carry out messages to final destination side (receiver).

Another constructive contribution to this approach is by Fragmenting large size of XML data file into customized smaller sizes/parts (chunks), encrypting it, and send the fragmented/shredded chunks each piece into different paths through cloud infrastructure via distributed RIDX instances, until all parts (chunks) merged back and decrypted in the destination side. This method will utilize the network resources for better performance by avoiding a network bottle neck problems, and/or dataflow problems that might arise (this usually happen when sender is faster than the receiver); therefore a dataflow control techniques has been utilized to solve this existing problem. In addition, shredding and multi path techniques add another line of defense in case of tapping the data by a Man-In-the-middle.

GENERAL WORK PROCESS OF RIDX

• **Initiation of Process**: When requesting to transfer XML data initiated by data management system or by any other interfaces integrated to the RIDX system, and after knowing the destination target for the XML file to be delivered, the Shredder/joiner module will parse the XML file and shred it into several chunks, each chunk will be read and encapsulated in a message, this message contain all the information needed to reach to the destination target. All chunks will be read until all data in XML file is processed. Meanwhile, the ready messages processed in the system will be prepared to be sent, therefore, the system will send a heart beat to the destination target to make sure that the server is up and ready to receive the data transmission. Also a list of potential servers will be prepared and listed from the data management system, the system will send a heart beat to all potential systems to check each server which has been chosen as a candidate routing server is up and ready to receive data transmission. Servers other than the destination target will act as a routing server. All servers that receives heart beat should respond, or else will consider this server is down and will be eliminated from the potential candidates and from the list.

After filtering the servers and check the readiness of all parties to receive data, the RIDX communicator start sending the prepared messages passed from the shredder module to the different parties randomly until finishes transmitting of the full XML document.

A session will be established between the sender and the receiver until the end of transmission of the message, the RIDX communicator will pass the received message to the joiner module. The joiner module will stack the messages until it receives all the messages. A checksum will be made for each message received to check the integrity and safety of the messages. When receiving all messages, the joiner module will rejoin all the messages to build the XML file same as the original one, another validation check will be done to check again the integrity and safety of the combined XML file. The joiner module will inform the RIDX communicator to send a confirmation message to the source sender and then close the session.

Exceptions could happen during this whole process, these exceptions will be handled case by case in scenario that guarantees the safety and integrity of the XML data file.

• Methods of Transfer

At this stage, adapting one method for transferring data is preferable until this system reaches to a point of maturity, and then it can be integrated to other methods as needed. A major method which RIDX communicator will depend on is Point-to-Point communication protocol, a TCP/IP socket communication network is the backbone of the communication between any two nodes, the module which is responsible of handling this kind of communication is RIDX communicator module. Using java technology as a tool to create environment contains metamorphism and multi threaded socket handling, this will guarantee the communication in multi session environment at the same period of time in an efficient way.

The backbone of the communication carriers can be used to facilitate the communication between all RIDX parties, a leased line, an ATM connection, or thru internet connection are all possible carriers. In addition, a various security protocols such as Virtual Private Network is also adaptable, the decision of choosing the carrier or choosing the security protocol depends on the firm.

• Transport

The RIDX will include the following for transport methods:

1. RIDX communicator guarantee the delivery of the XML files
2. RIDX Shredder/Joiner guarantee the Integrity of a record level along with the file level, a checksum or MD5 techniques is used
3. Data encapsulation used to guarantee privacy of data
4. RIDX combines reliability and multicasting
5. Support for all file types binary, text etc. (Future Design)

Topology, Architecture & Components:

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. Each RIDX system contain modules, processes and listeners , some responsible for keeping track of domain members, some for delivering and receiving messages, some for routing messages and others for shredding/joining messages… etc.

- Topology

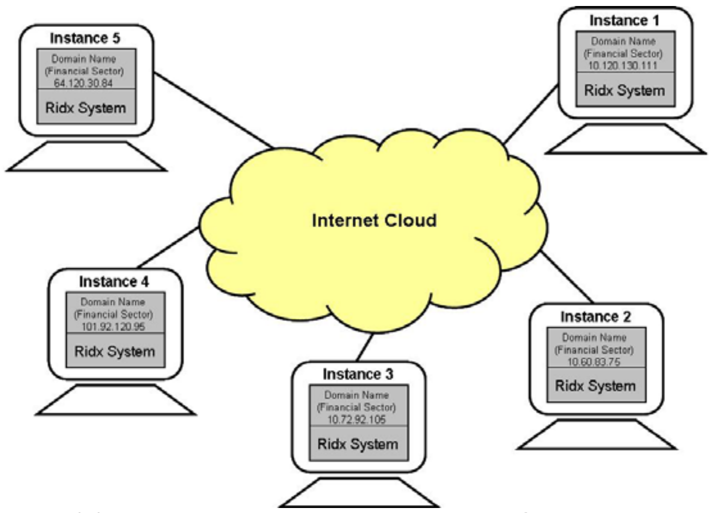


Figure (1), RIDX Instances distributed in the Cloud, Domain interest is “Financial Sector”

The RIDX architecture divided into two parts:

1- Pack of modules containing classes to implement XML data exchange between different RIDX systems on the cloud platform

2- The interface (API) which provides the accessibility to different types of applications to enable them to use RIDX as middleware for XML secured data exchange between all parties.

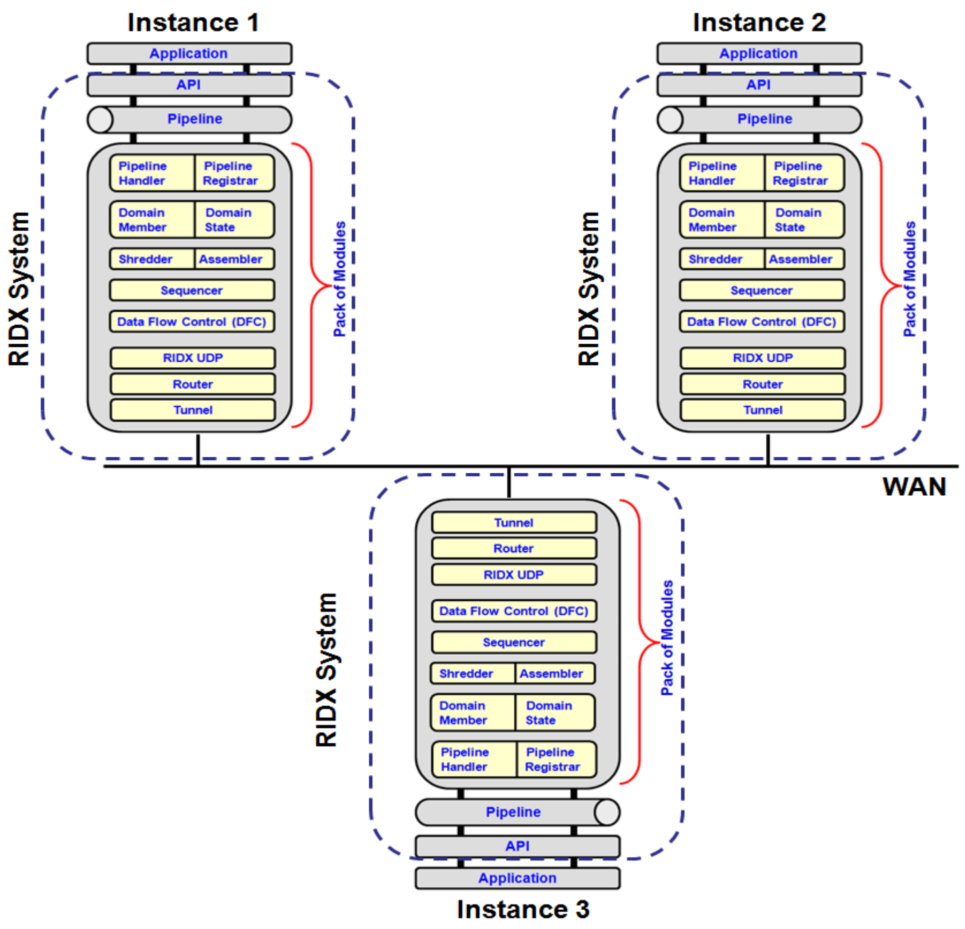


Figure (2), RIDX Instance structure & different modules which are used to facilitate major functionality

An Overview of RIDX Architecture:

Real-time Interactive Data Exchange system (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 interest between members. 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 leave 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 the domain by creating a process with the same domain name. The handler between domain members is the PIPELINE; this means 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 UDP transport protocol put the processed data on to the network.

In the reverse order, when the destination receives the data, the RIDX UDP transport protocol listens to incoming data via 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 guzzle 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.

The Conceptual Model Design & Components:

As shown in figure (2), a pack of modules facilitate the functionality of the system; the Application Programming Interface (API) is the interface between RIDX system and the software accessing it to allow using various services and resources, along with different modules and protocols. Each XML file sent by application/user through pipeline/API should be processed by modules until it reaches to the lowest layer (UDP layer or Tunneling layer). All messages received by communication layer should be processed by the pack of modules up until it reaches the application layer. All RIDX system parameters are saved in XML format, which define the desired outcomes from the system.
The following modules describe the major functionality used by the system:

- **Listeners/interfaces:**

  - 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.
  - 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 distribute it after register the new status to all domain members. The domain coordinator is in charge of accepting/leaving and announcing the new snapshot.
  - Pipeline registrar: In order to send and receive files, the user need to register in a pipeline, the pipeline name identifies the domain and the coordinator holds the state of the pipeline whether is active, inactive or closed.
  - Pipeline Handler: The pipeline handler passes all XML files from the application to the pack of modules or from pack of modules to the application.

- **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 desired to run more than one RIDX instance on the same node, then all instances has the same identifier (IP Address) except for the port number. Therefore port numbers differentiate between multiple instances on the same node.

- **RIDX Message:**

  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. The message structure is as follows:

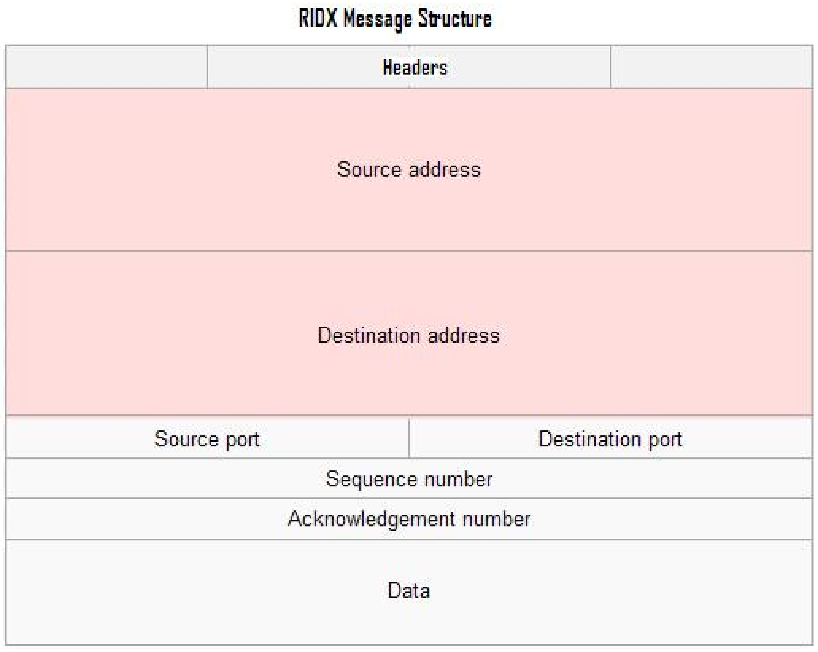


Figure (3), RIDX UDP message structure used in transport protocol layer

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, other method is to detach/delete, and so on. Headers are attached whenever data should not be put in the Data Field.
- **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. Data can be encrypted and compressed if needed.
- **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 to identify each segment of the data.
- **Acknowledgement Number**: RIDX is a Multicast reliable delivery, therefore message acknowledgement is guarantee delivery of all messages, in case of lost packets/messages, and the source will retransmit the lost packets.

- **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 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). In the same sense, whenever a domain member leaves the domain, the coordinator will produce a new AML and all domain members will notice the new AML sequence.

- **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 pipelines is active and in connect state, it is always allied with a particular domain. Pack of modules look out that pipelines that are holding the same domain name can find each other. When a new joiner joins a domain, a new AML (Active Member List) will be generated and distributed to all members using domain members' listener.

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:

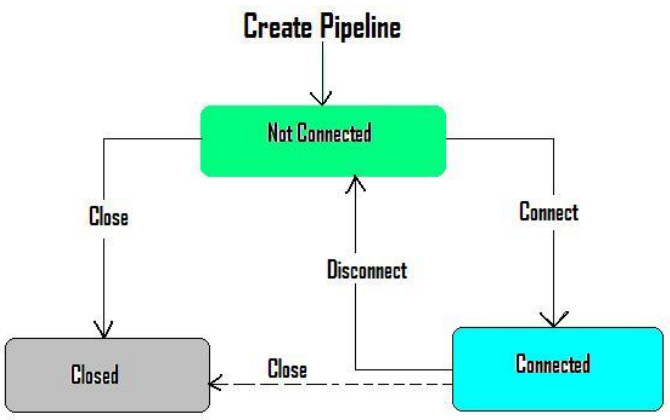


Figure (4), Different states of the pipeline & the process of changing between them

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.

Modules & General Work RIDX:

- **Initiation of Process:**

When requesting to transfer XML data initiated by user 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 belongs 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 configuration file in the RIDX XML configuration file. Each piece is encrypted and encapsulated in RIDX message structure, each RIDX message contains all the information needed to be transferred until it reaches to the final destination. The system will choose different path for each chunk according to the Active Members List and transmit simultaneously in a TCP multicast reliable transmission:

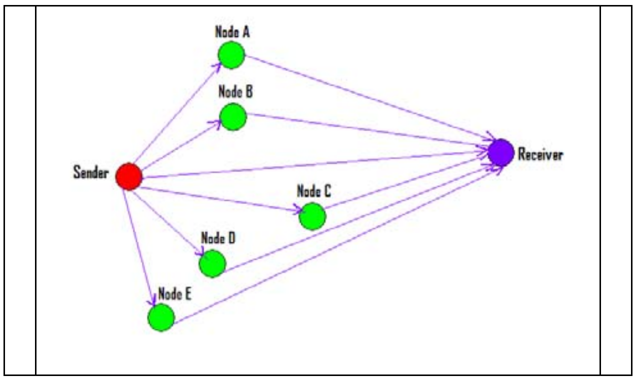


Figure (5), RIDX Internet Cloud Domain Members: Domain Name "ABC"

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

- **Shredder & Assembler**

The shredder & assembler modules are one of the defense mechanism of this architecture. The purpose of these modules first is to break down the XML file into several chunks (small pieces), adding an identifier to each chunk, then attach this identifier to a message along the data; the id can be added separately either by adding a header to the message or embedded in the message. At the final destination (Receiver), all messages are re-assembled again.

The size of the chunk is pre determined in the configuration file, this decision depends on what level of security needed to be applied. For example if 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, but it has several drawbacks:

- o Lower performance and inefficiency when reassembling fragmented data
- o Lower performance and high cost of retransmitting when number of lost fragments is high
- o In standard UDP case, Lower performance 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, route and Operating system been used.

By considering these disadvantages, Shredder/Assembler & RIDX UDP modules were built to overcome these problems.

- **Data Flow Control**

Data Flow Control is responsible to control the speed of transmitting when the receiver is slower than the sender. When messages overflowed 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. unfortunately none of these techniques can be deployed in RIDX communication structure, because these techniques assume a Unicast one-to-one single connection and it will not work within RIDX architecture. Therefore another approach is 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.

As a result, 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. This module uses a similar to an acclaim bank account transaction, which is whenever the sender sends a message, the sender decrements from a maximum pre defined 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 message, it reimburse the sender bank account value by 1. Messages in the receiver side will be processed and queued in the pipeline until all messages is received and reconstructed back to its original XML file structure.

- **RIDX UDP Transport Layer**

The RIDX UDP transport layer employs IP Multicast with an added feature of multicast simultaneous transmission functionality, in addition; it is reliable transmission by adapting ACK-NACK (Acknowledgment & Negative Acknowledgment) message control mechanism, which provides First in First out (FIFO) guaranteed message delivery transmission.

Choosing the best and appropriate network infrastructure model is an important step to setup the network between different partners. Nowadays, different scenarios are available; such as leased lines, Point-To-Point VPN, and Internet cloud, all these types are very common, but each option has features with advantages and disadvantages. Next section will shed light on these advantages and disadvantages and to give a view for about where RIDX system can deploy RIDX system according to the available communication infrastructure.

Briefly, a summary of the different network infrastructure that RIDX can be adapted to, and a comparison between network types with its advantages and disadvantages:

- **The Internet Cloud**

Internet cloud communication is admittance over an unrestricted access available to public. Secured and reliable communication environment is a big challenge for almost all businesses operates using internet cloud. Nevertheless, internet cloud provides a cost-effective platform for different businesses, services and type of communication to choose.

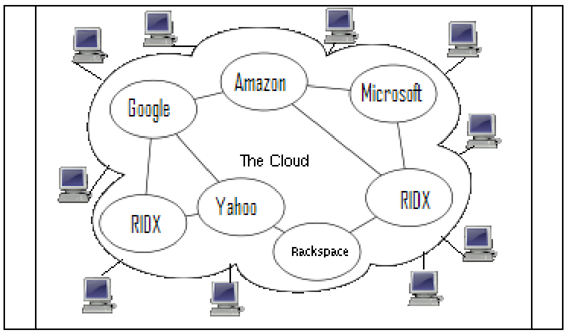


Figure (6), RIDX Instances within the internet cloud

RIDX system virtually provide an environment to carry out secured communication between different services and transferring XML data, it can also work as SaaS (Software as a Service)

- **Leased Lines**

Normally, organizations use leased lines to connect far-off geographical locations, for a fixed monthly rate, the distance between connected locations is the major factor of calculating the monthly rate, the more far is the distance the higher rate is charged. The cost of this type of connection is a big consideration.

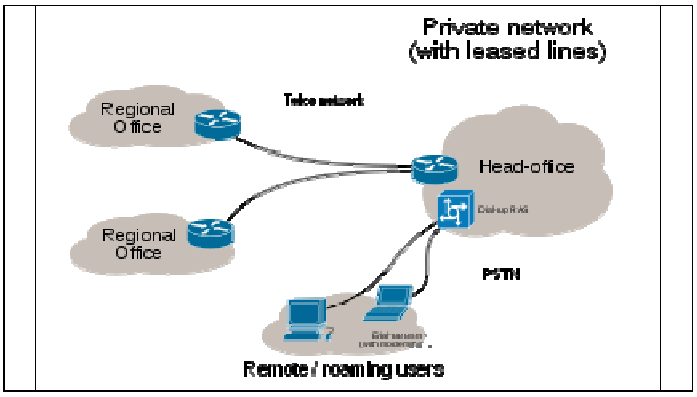


Figure (7) shows direct connect thru Leased lines

- **POINT-TO-POINT VPN**

Virtual Private Network is a way of securing the communication between two parties by crypto-graphing and tunneling the connection, it encapsulates the data by using cryptographic methods. Only authenticated users can access this line, therefore, unauthorized users will be blocked of using this connection.

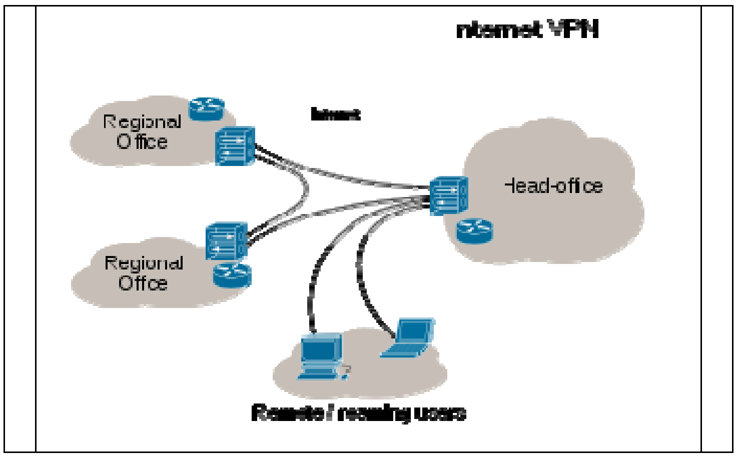


Figure (8) shows the connect thru P2P VPN Connection

Understanding the advantages and disadvantages of each type of communication will clear the vision on what type connection should be used or maybe combination of more than one type can be utilized to serve the purpose of this project. Hence, a list of advantages & disadvantages is presented to be considered as part of the decision to study the topology of the connectivity design:

Network Type	**Advantages**	**Disadvantages**
Leased Lines	Leased lines are attractive solution, because it provides private physical connection between different nodes with a total privacy.	Building leased line network connection between all nodes / partners is not an easy job; it is a bit complex management wise and high cost, especially if the number of nodes gets high.
Internet cloud	The Internet can hold large number of participants of different kinds of networks, it is also a scalable solution in terms of connectivity, and has a considerable cost saving resulting from its multiple uses. Dissimilar leased lines, connecting thru Internet can work without a dedicated hardware to be used for each installation, therefore set it up is much quicker. In addition, a large number of users of any network can connect with set-up cost considerable.	No control over the Internet in terms of security. For instance, Although comparatively cheap to deploy, the opportunity of failing down the connectivity. The Internet is also insecure. On the other hand, to utilize encryption mechanism such as SSL, Tunneling, DES, or PGP is one of the options to overcome this. Some firms still have concerns as a strategy of not to use the Internet for sensitive business data exchange. Thus internet connectivity solution may present harms to some essential business partners.
P2P VPN	The major advantage of tunneling by using point-to-point/VPN is reducing a complexity to manage multiple connections. In spite of number of connections to business associates using VPN, only required to administer a one physical line connected to the network. Setting up a new connection is basic and only a logical activity; if one connection fails all your remaining connections is unaffected. VPN connections generally very fast.	If the business partners is linked to the network you choose, that is apparently a major weakness. Also, VPN connections that function over public networks can face a risk of loosing priority to other types of traffic. Moreover, if losing the physical connection, there will be loss to all logical connection using this physical connection.

• Tunneling

One of the obstacles that can face RIDX is to conduct a proper communication when RIDX instance resides behind a firewall. Due to a great risk using public internet communication, organizations use a firewall to protect the organization against attacks, and impose access control. for that reason, mostly firewalls prevent the connection from outside to inside, but allow some inside applications to connect outside services and machines. Tunneling is a protocol that encapsulates other protocols, and provides a mechanism to allow RIDX to operate through firewall systems and security gateways:

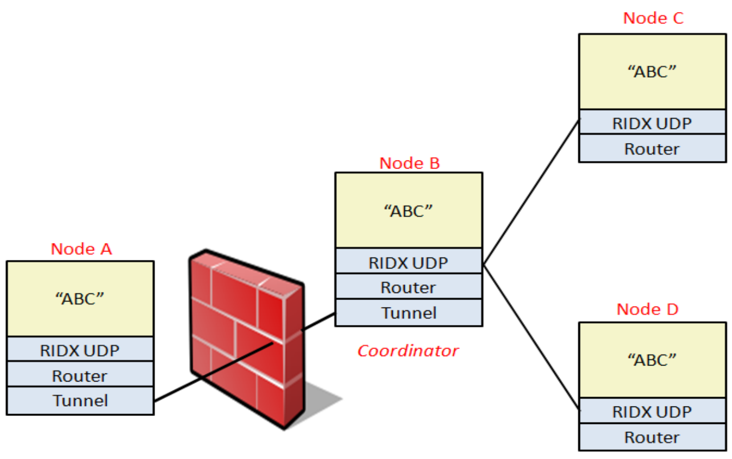


Figure (9), Creating a tunnel between a Node behind the firewall and a Node in the internet cloud

The question we need to ask: "Is it feasible to implement VPN tunneling between all RIDX instances?" in order to answer this question, we need to look at different aspects when applying it:

First, implementing tunneling mostly can be expensive due to the need of different technologies involved, such as Hardware, software and security network specialists to implement and maintain it.

Second, when hosting RIDX instances internally in the intranet organization, tunneling is not applied, and still the transferred data can be exposed by Man-in-the-Hub.

Third, 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 tunneling on different data frame sizes. Traffic generated at 100% of the line rate of (1000 Mbps). Results showed:

Frame Size	Throughput BEFORE VPN Tunnel	Throughput AFTER VPN Tunnel
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 (2), Throughput Data of VPN (Before and after)

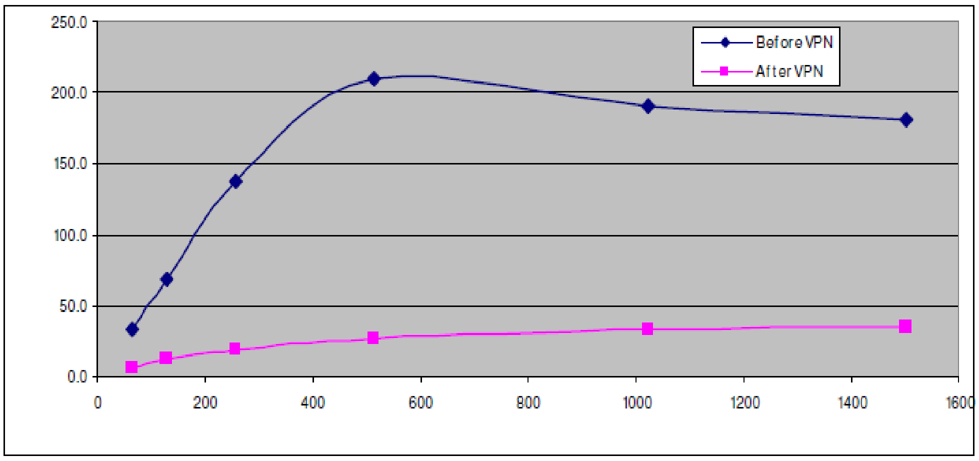


Figure (10) Throughput Data of VPN tunneling (Before and After)

A delay has been measured when tunneling configuration done, time is measured in microseconds for 1 packet traffic per second. Next table shows the result:

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

Table (3), before and after VPN one-way tunneling

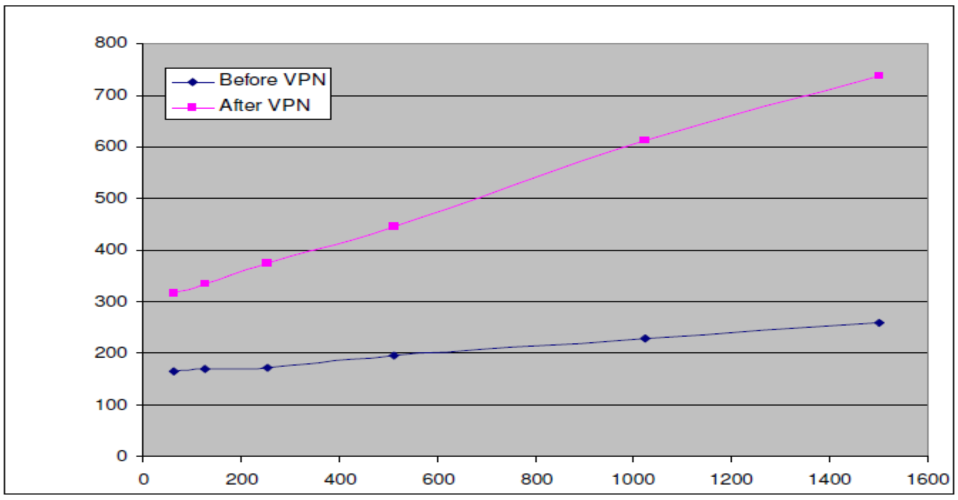


Figure (11), before and after VPN one-way tunneling

As a result of this experiment, using tunneling technique between all RIDX nodes is not the best option, because the results showed that there is considerable latency when using tunneling techniques, consequently, choosing tunneling only in certain situations is acceptable where RIDX resides behind a firewall and has limited access in and out side to the RIDX cloud. A module TUNNEL is responsible of 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 tunneling.

As in figure (9), the initial process of conducting a secured tunnel starts from (Node A) which is assumable behind the firewall, requests to join a domain from the coordinator, the coordinator accepts the new joiner, Node A establish 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 will fail, therefore Node B will act as a router for all message to/from Node A.

Nevertheless, tunneling technique in RIDX has several downsides. First, using Node B as a dedicated router might exhaust and consume the resources from Node B. second, firewalls owned by a third party not always possible to control the desired enabled ports needed to operate. These two reasons could lead to scalability issues.

RIDX Performance Test

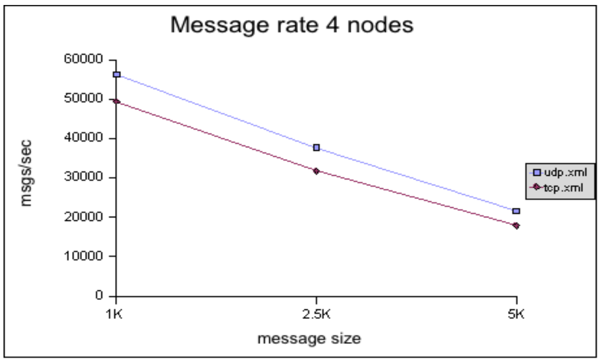
A test conducted on Dell 1850 3 GHz with 4GB RAM on 2.6.9-14 RedHat Enterprise Linux 4 / 64 bit system, a Dell switch of 1GBps, SUN JVM 1.6.0_3. Each node (box) contains RIDX system, it will send number of M messages to all cloud nodes; receiver can be a sender at the same time, the timer will start when first message has been received, again the timer will stop when last message has been 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 will be calculated by multiplying {number of senders N} times {number of messages M}. A configuration file contains same parameters for all RIDX nodes:

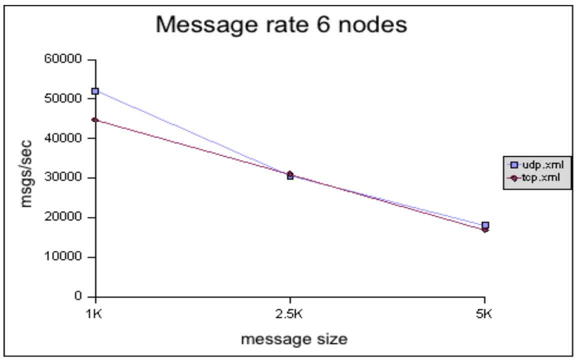
- N = 4 then 6 then 8, senders are same receivers, that means N equals to number of RIDX nodes also
- M = 1,000,000
- S = 1 then 2.5 then 5 KB the size of each message

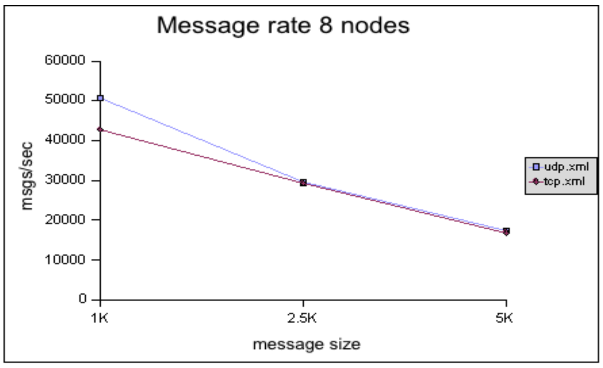
RESULTS

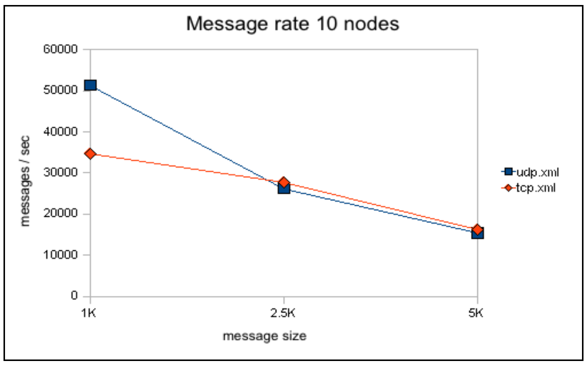
The computation of message rate = (M * N) / T, where T is the total time of receiving all messages. Therefore, X & Y axis represents (S) and (Rate & throughput) simultaneously. For example: a collection of data results when the test is done, we need to calculate the average rate of number of messages 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). As shown below in the graphs.

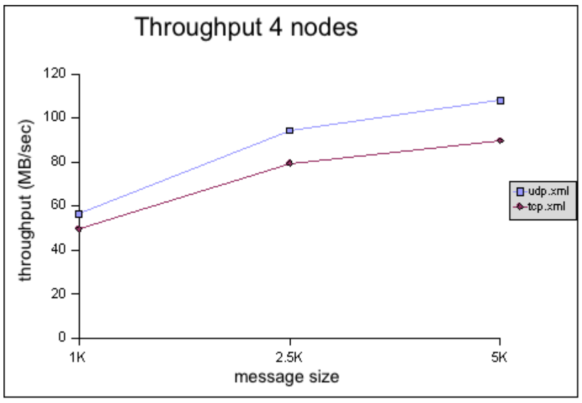
When compared standard TCP against RIDX UDP, a TCP & UDP variance between four to ten nodes, we notice that the TCP begins at 49 MB per Second until it drops down to 34 MB per second, on the other hand, UDP begins at 56 MB per Second until it drops down to 51 MB per second, this shows that a significant 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

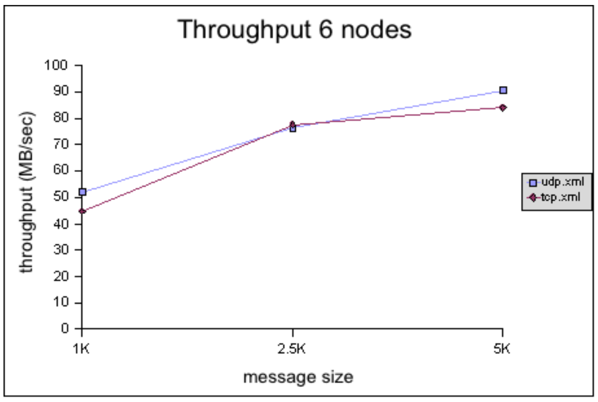


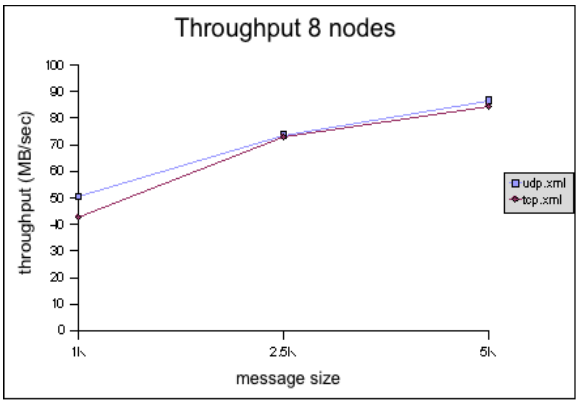


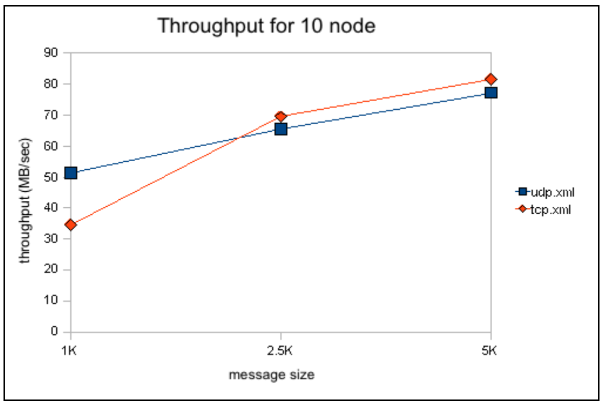

Message rate 8 nodes
msgs/sec
message size
udp.xml
tcp.xml

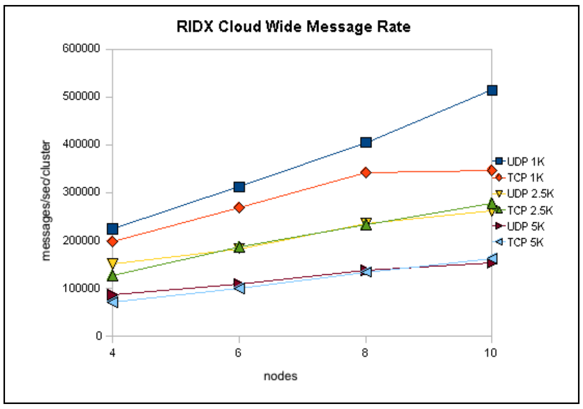

Message rate 10 nodes
messages / sec
message size
udp.xml
tcp.xml

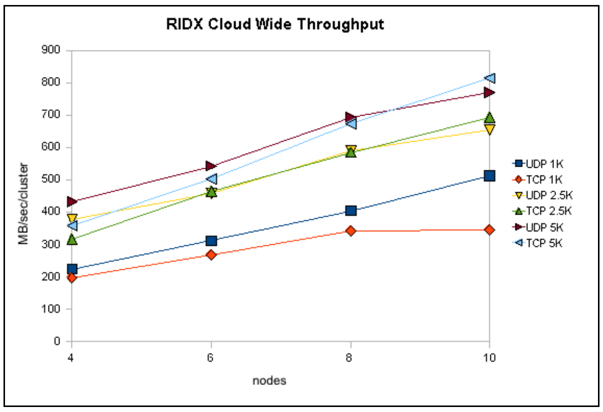

Throughput 4 nodes
throughput (MB/sec)
message size
udp.xml
tcp.xml


Throughput 6 nodes
throughput (MB/sec)
message size
udp.xml
tcp.xml


Throughput 8 nodes
throughput (MB/sec)
message size
udp.xml
tcp.xml


Throughput for 10 node
throughput (MB/sec)
message size
udp.xml
tcp.xml


RIDX Cloud Wide Message Rate
messages/sec/cluster
nodes
UDP 1K
TCP 1K
UDP 2.5K
TCP 2.5K
UDP 5K
TCP 5K


RIDX Cloud Wide Throughput
MB/sec/cluster
nodes
UDP 1K
TCP 1K
UDP 2.5K
TCP 2.5K
UDP 5K
TCP 5K

DISCUSSION

Security data threats are always subject under improvement & development. Secured XML file 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 a multilevel line of defense for securing XML data during transmission between different organizations, businesses and services.

The experimental test is made to measure the performance, putting into consideration 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 reliability delivery is an achievement to this approach.

REFERENCES

Arnab, A. & Hutchion, A. (2005). Requirement analysis of enterprise DRM systems. In Proceedings information security South Africa, Hotel Balalaika, Sandton, Johannesburg. http://pubs.cs.uct.ac.za/archive/00000205/01 Accessed [2008].

American Mathematical Society, [2009]. "Advances in network information theory". Providence. pp. 321–334. ISBN 0-8218-3467-3. Retrieved

Arnold, Jon. [2010] "Why businesses need to think differently about cloud communications"

Atwood, J William & Barhoush, Malek, [2010]. Telecommunication Systems Vol: 45 Issue: 1 ISSN: 1018-4864 Page 3-20: Digital rights management (DRM),

Biddle, P., England, P., Peinado, M., & Willman, B. The darknet and the future of content distribution. http://msl1.mit.edu/ESD10/docs/darknet5.pdf . Accessed [2008].

Bryant, E. D., Atallah, M. J., & Stytz, M. 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 [2008].

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

Blahut, Richard E. [2004], "Algebraic Codes for Data Transmission" Cambridge University Press, , ISBN 0521553741.

Comer, Douglas E. [1995], "Internetworking with TCP/IP, Volume 1: Principles, Protocols, and Architecture", Prentice Hall, ISBN 0132169878

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

Carlos Serrão, Miguel Dias and Jaime Delgado, [2006], Bringing DRM Interoperability to Digital Content Rendering Applications, 323-330, DOI: 10.1007/1-4020-5261-8_50

Christopher A. Kent, Jeffrey C. Mogul. [2000], "Fragmentation Considered Harmful"

Comer, Douglas E. [2006], Internetworking with TCP/IP: Principles, Protocols, and Architecture. 1 (5th ed.). Prentice Hall. ISBN 0131876716

Don Murphy and Donal Heffernan. [2003], Assembly Automation Vol: 23 Issue: 1 ISSN: 0144-5154 Pages: 60 - 68.

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

Dan Schaffer. [2011] "Network security in the Automation world ", InTech Vol: 58 Issue: 2 ISSN: 0192-303X,P. 58

Felten, E. W., & Halderman, J. A. Rootkits, [2006], “digital rights management, spyware, and security”. Los Alamitos: IEEE Comput.Soc

FIX Protocol Ltd. FIXML, [2010]. A markup language for the FIX application message layer. Version 4.4. http://www.fixprotocol.org/specifications/fix4.4fixml

FIX Protocol Ltd. [2009].The Financial Information Exchange Protocol (FIX), version 5.0, http://www.fixprotocol.org/specifications/FIX.5.0SP2

FIXML – FpML, [2010]. Overview of the schema FIXML 4.4, http://www.fixprotocol.org/specifications/fix4.4fixml

F. E. Bustamante, G. Eisenhauer, K. Schwan, and P. Widener, [2000]. Efficient wire formats for high performance computing. In Proceedings of the 2000 Conference on Supercomputing, Author, Gont, Fernando (2008-11). "On the implementation of TCP urgent data". 73rd IETF meeting. Retrieved [2009]

Gregorio J. URI Templates. [2009], http://bitworking.org/projects/URI-Templates

Grötschel, Martin; Alevras, Dimitris; & Wessäly, Roland, [1998], "Cost-efficient network synthesis from leased lines",Annals of Operations Research Vol: 76 Issue: 0 ISSN: 0254-5330, Pages: 1-20

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

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 [2011].

Ho, Zuleita Ka Ming; Lau, Vincent K. N; & Cheng, Roger S. K, [2009], "Cross-Layer Design of FDD-OFDM Systems based on ACK/NAK Feedbacks"

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

IBM:z/OS operating system". 03.ibm.com. [2009] http://www03.ibm.com/servers/eserver/zseries/announce/zos_r4/ Retrieved

IEEE, IPv6 and Broadband ISBN 3-00-013801, [2005]

Jianyong Chen; Cunying Hu; & Zhen Ji, [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

Jon L Jacobi, [2011], Optimize Your Router for VoIP and Video Streams , PC World Vol: 29 Issue: 3 ISSN: 0737-8939, Start Page: 89

J. Arturo Pérez, et. Al. [2006]. "Quality of Service Analysis of IPSec VPNs for Voice and Video Traffic," Advanced International Conference on Telecommunications, Guadeloupe

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

Li, Harnes, Holte, [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

Kohloff, Christopher and Steele, Robert: Evaluating SOAP for High Performance Business Applications: Real-Time Trading Systems, [2003], Systems, 2003, http://www2003.org/cdrom/papers/alternate/P872/p872\kohlhoff.html, accessed [2005].

Mazurczyk, Wojciech and Szczypiorski, Krzysztof, [2009],“Cryptography and Security”

M. Govindaraju, A. Slominski, V. Choppella, R. Bramley, and D. Gannon. [2000]. Requirements for and evaluation of RMI protocols for scientific computing. In Proceedings of the 2000 Conference on Supercomputing,

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

Nicholas Rosasco and David Larochelle. [2006]. "How and Why More Secure Technologies Succeed in Legacy Markets: Lessons from the Success of SSH" (2001). Dept. of Computer Science, Univ. of Virginia. Accessed

Pautasso, Cesare; Zimmermann, Olaf; Leymann, Frank (2008-04), [2008], "RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision" (HTML), 17th International World Wide Web Conference (Beijing, China)

Peterson, Larry L. & Davie, Bruce S. [2000], "Computer Networks: A Systems Approach", Morgan Kaufmann, ISBN 15586051428888.

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

R. Wagner, et. Al. , [2006], “Recommendations for Infrastructure Protection, 2006,” Gartner Group

Rittinghouse, John W, [2009], "Cloud Computing : Implementation, Management, and Security", Edition: 1 Call Num: 004.36 ISBN: 1439806802, Pages: 340.

Rabhi, F.A. & , Benatallah, B. [2002]: 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

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

Soljanin, Emina; Ruoheng Liu & Predrag Spasojevic, [2004], "Hybrid ARQ with Random Transmission Assignments".

Tomas Olovsson, Wolfgang John, [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.

Whitemore, James, [2009]. "Unified communications for the enterprise: hosted VoIP or cloud communications".

Wilkinson, Paul, [2005], Construction Collaboration Technologies: The Extranet Evolution. Taylor & Francis. ISBN 0-415-35859-0.

Thornycroft, Peter. "Moving communications into the cloud", [2010]

zur Muehlen, M.; Nickerson, J.V.; Swenson, K.D. [2005]: Developing Web Services Choreography Standards – The Case of REST vs. SOAP. Decision Support Systems 37, Elsevier, North Holland.