IJCSN International Journal of Computer Science and Network, Volume 3, Issue 5, October 2014
ISSN (Online) : 2277-5420 **www.IJCSN.org**
Impact Factor: 0.274

Dynamic Port Generation Using Any Communication Protocol over Custom Security Layer

[1] Siva Srinivasa Rao Mothukuri

[1] Information Technology, RVR & JC, Acharya Nagarjuna University
Guntur, Andhra Pradesh, India

Abstract - Now-a-days, with the increase in the use of resources over a network, many people are failing to access their required resources. This is caused due to unavailability of ports. Unfortunately, reserved ports can be accessed only when the ports are free. If there is a communication link, some ports will regret to accept any new incoming connections and we need to wait until other connection closes its established link. To overcome this problem, client-server architecture is meliorated to certain standards for having dynamic ports. This also uses a table for storing of connections for further reference. In addition to that, a custom security layer is introduced for protection of data from an attacker. This becomes most useful when there is heavy network where network resources are not adequate.

Keywords - *AES, DES, Client-Server Architecture*

1. Introduction

Communication is a base for every hardware units to interact with other hardware. There are several ways for hardware to communicate. However, base concepts remains the same i.e., opening a socket, establishing a handshake with other equipment then exchanging of messages. As already stated, this communication can be achieved in several ways like IPC, Sockets but we are not going to deals with this.

We know that Client Server Architecture that access port. What if we try to connect to same port which has already established a connection with hardware equipment? The result is as expected; we are unable to connect to that port as it is already in use. The main idea behind this paper is to make dynamic port generation that uses any protocol. If we consider same example, when port is busy, then any unreserved random port is chosen and same process of communication is continued.

2. Dynamic Port Generation

Dynamic port generation is a simple method to generate ports that uses any protocol for communication. For this, we need to follow similar structuring.

1. Server instance/port generation
2. Client port generation

2.1. Server Instance / Port Generation

For a communication between two hardware devices, Server Client architecture is most commonly used as it is simple to implement and robust. This architecture opens a port and establishes connection between two ports.

Here, server port generation is dynamic. It checks for unreserved ports, and then opens a port which is free. Conundrum is how client knows about this port. This will be discussed in later part of this paper. Dynamic property of instance generation comes into picture when any client tries to connect to the same port. Usually, server returns a message saying “unable to connect”. However, this is not same case; a new instance is created for the same port then that can be used based on the concept of multi-threading.

So, what happens when we exceed the limits of instances for the same port? Now dynamic port generation occupies a special role in searching new port and generation of free port. Everything is recorded with a table just as routing table. It records all required information such as client ip, port, mac, hop.

Table 1: Illustrating the drawback and advantages

Name	**Drawback**	**Advantages**
Normal port (client server architecture)	No other client can use the same port.	Securely connects to one client only
Dynamic instance generation	Limit for instances	Creates a new instance for other client to same port

		using threads
Dynamic port generation	Limit for Ports (Hardware)	Creates a new port, if required and also if all instances are in use

To discuss about establishment, this paper is also concerned about the security layer. Since any client can connect to the server a new layer is added on top of communication. Every message passes through this layer following custom encryption and decryption standards. There is a property file which states the type of security that can be used and levels of security. In addition to that, we can add random salt not just by having simple encryption hierarchy.

Snapshot of property

 # Author: Siva Srinivasa Rao Mothukuri
First_Level_Security="DES"
*Random_Salt_First="ILiN*e@3xdG2$"*
Second_Level_Security="AES"
Random_Salt_First="W2(mx&iEk^"*

To start the communication, client sends set of messages to have initial handshake which is in encrypted format. If this fails it will never connect to that open port.

2.2. Client Port Generation

Considering client perspective, client doesn't know which port is used by the server for communication. This becomes hectic problem for client. However, when client finds an open port, it sends set of initial encrypted handshake requests. Once authentication is finalized then client communicates with server over custom encryption layer.

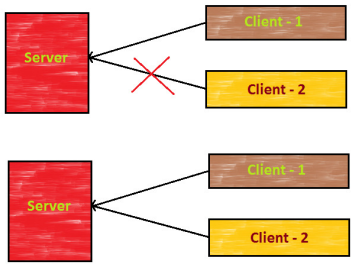


Figure (a) delineates that client 2 is unable to connect to server with normal port. But in figure (b) client 2 is able to connect. encryption and decryption over unsecure medium

Bellow figure shows architectural flow of messages over unsecured medium like internet. As we have custom security, this becomes much harder for an attacker to intrude into the messages. Even though if he captures messages, he will not be able to read the transactions that are sent. This enhances the security for messages.

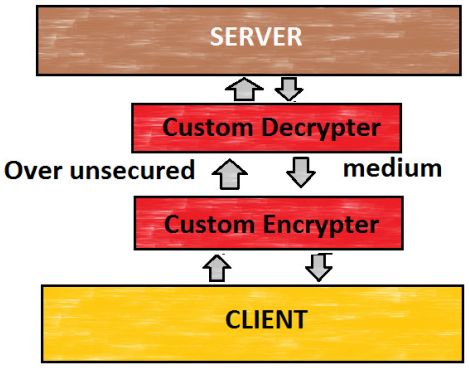


Figure b. delineates architecture of custom

3. Custom Security Standards

Now, this is the time to discuss where security must be taken into consideration. As we are using unreserved ports for connection, there is huge probability for an attacker to intrude or capture all the activities that are ongoing through the port. So, if there is no security then entire architecture will be in vain.

To implement this, we have introduced two layers in-between the client and server for exchange of messages over unsecured medium. Even if any attacker tries to sniff the data, entire data is in encrypted format. This results no use even if there is man-in-middle attack.

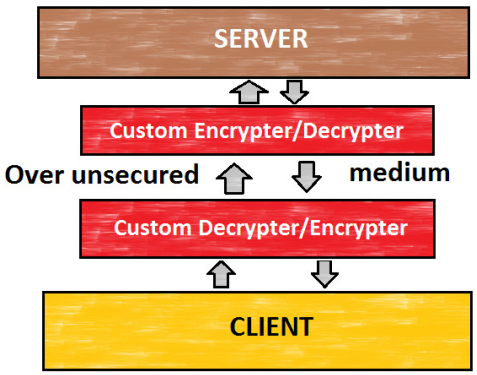


Figure c. illustrating encryption and decryption vice-verse

The encryption decryption standards can be managed through a property file where there is multilevel support for security along with random salt.

4. Implementation

The implementation of this architecture is moderate. This also depends on worst case scenarios like searching for ports. So, hardware of server and client must be capable.

4.1. Algorithm for Searching of Freely Available Ports

Algorithm :: Search_Function
Input :: Range of unregistered ports
Output :: Get free port to start communication

Function Search_Function
Start:
For range of ports given as input
If port is already in use or open
Leave the port and continue to search
Elseif found a free port
Return to next level
Else
Throw exception-No ports available
Endif
End

4.2. Algorithm for Server port generation

Algorithm :: Server_port_generation
Preconditions :: A port must be available
Input :: Port or client trying to connect
Output :: Create a new port and start to have communication

Function Server_port_generation
Start:
If port is available
Open the port and start to create service
Elseif client is trying to connect to a port
Create a new port if no instance is present
Endif
Record all details into table for further reference
End

4.3. Algorithm for Client Port Generation

Algorithm: Client_Port_Generation
Input :: Port to connect to server
Output :: Establish connection to server

Function Client_Port_Generation
Start:
Start to search for the port that has server connection
If port has similar encryption standards
Establish connection with the server
Complete handshake
Exchange messages between server and client
Endif
End

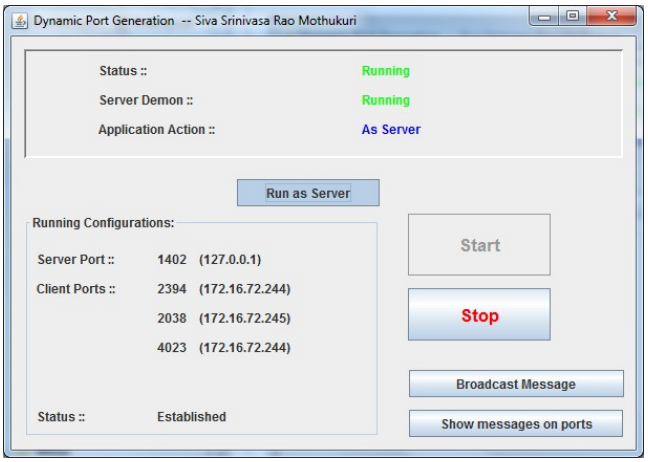
4.4. Algorithm for Encryption/Decryption

Algorithm :: Encryption_Decryption
Input :: Establishing connection between Client and Server
Output :: Secured messages

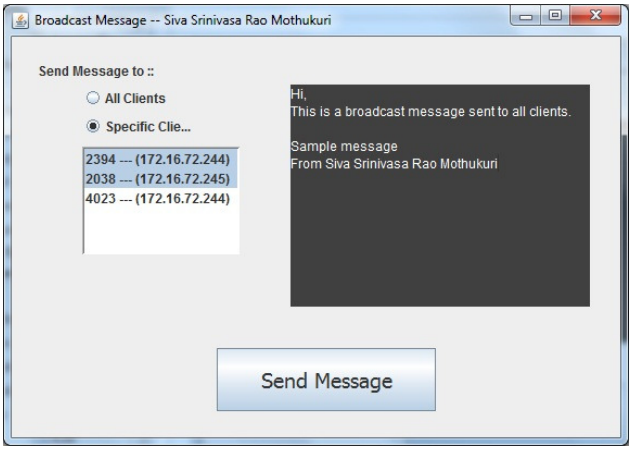
Function Encryption_Decryption
Start:
Get all the details from the property file
For every layer of security
Follow the security method specified in property
Add random salt for the before encryption
Follow same for next level of security
Done
End

A sample snapshot of the work is shown below.

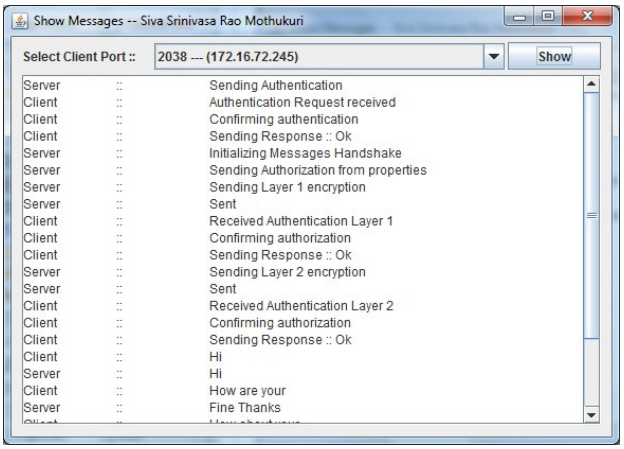
Here the application is switched to server mode. So, the application is able to connect to the clients through dynamic port.



Server can send broadcast messages to all its clients. Through the option “Broadcast Message”. There is an option to send messages to particular specific clients.



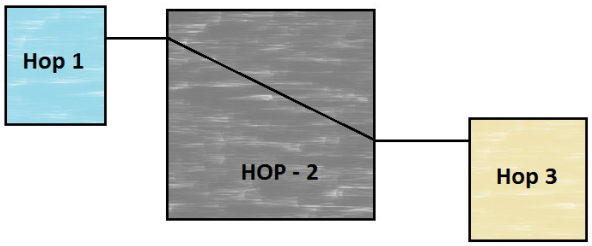
Likewise, the application is able to see all the conversations that are happening between selective client and the server.



The same is shown for the client. However, there is a restriction that client can only view its own connection configuration rather that all client conversation.

5. Further Improvements

We take proposed architecture to next level. By having similar architecture in a distributed network, every hardware unit works individually and creates a new port if same port is not available. To be in-specific, whenever a hardware uses a port to communicate with other neighbor hardware, same port might not be used for other hardware to communicate. In fact, hardware can dynamically switch the port if required. This will increase re-usability and supports more number of connections.



In this figure middle hop uses its connection with different ports with an internal connection in-between.

5. Conclusions

Reserved ports can be accessed only when there is no connection. If there is connection, some ports will regret to accept the connection and we need to wait until other connection closes its link. In this paper, we discussed how a random port can be used for communication and further how it can be used in distributed network where availability of ports are very less. This architecture suites best for large scale networks where there is limited ports with huge number of resources waiting for connection.

References

[1] Bertocco, M. Ferraris, F. ; Offelli, C. ; Parvis, M. ;"A client-server architecture for distributed measurement systems ","Instrumentation and Measurement, IEEE Transactions on (Volume:47 , Issue: 5) ", IEEE

[2] Abdul-Fatah, I. ;Majumdar, S.; "Performance of CORBA-based client-server architectures","Parallel and Distributed Systems, IEEE Transactions on (Volume:13 , Issue: 2) ",IEEE

[3] Patrick G. Bridges1, Member, IEEE, Gary T. Wong2, Matti Hiltunen3, Member, IEEE Computer Society,Richard D. Schlichting3, Fellow, IEEE, and Matthew J. Barrick1, Member, IEEE; "A Configurable and Extensible Transport Protocol",ACM/IEEE TON3

[4] Chih-Chung Lu ,Shau-Yin Tseng ;"Integrated design of AES (Advanced Encryption Standard) encrypter and decrypter ","Application-Specific Systems, Architectures and Processors, 2002. Proceedings. The IEEE International Conference"

Siva Srinivasa Rao Mothukuri born in Guntur, in the year 1991. Acquired bachelor's degree from Acharya Nagarjuna University in 2012, also Certified in Ethical Hacking. Worked as Assistant Professor in R.V.R & J.C College of Engineering. Currently working as ASE in Tata Consultancy Services (TCS).