Development Of An Enhanced Bandwidth Control Platform For Effective Monitoring And Utilization Of Resources In Corporate Networks

Chimezie Oji, Obi Nwankokwo, John-Otumu A.M, Christiana Adu

Department of Information Technology, Federal University of Technology, Owerri Imo State

DOI: 10.29322/IJSRP.11.08.2021.p11636
http://dx.doi.org/10.29322/IJSRP.11.08.2021.p11636

Abstract: An enhanced bandwidth control platform for effective monitoring of resource utilization in corporate networks was developed based on the issues identified with corporate networks such as: no effective control of the use of bandwidth, constant attack on core applications of the institution due to users surfing sites they ought not to, no sure way to pick out the device that consumes most of the resources, a constant increase in the cost of purchasing more bandwidth to meet the demands of customers, no effective way to identify devices housing applications and sites that are a threat to the institutions, no effective way of isolating the devices consuming the data, and no effective use of bandwidth to increase the general operation of a company’s corporate network. The Agile software methodology was adopted based on its merit in software development life cycle. Programming languages such Hypertext Markup Language 5, Hypertext Preprocessor, Cascading Style Sheets, JavaScript, JQuery, and C++ were used to translate the architectural blueprint into software product for live deployment. The proposed system has some major functional modules for screening of devices using protocol based techniques.

Keywords: Development, Bandwidth, Control System, Network, Resource Utilization

1. INTRODUCTION

Bandwidth is measured as the amount of data that can be transferred from one point to another within a network in a specific amount of time (Paessler, 2019). Recently, we've seen the use of bandwidth grown at a geometrical rate, which continually results in the extra budget allotted to the acquisition of extra bandwidth by most organizations. Analyses has shown that most resources that devour the bandwidth aren't associated or tied to the main function of the institution, with a large chunk of bandwidth utilized by sites such as Instagram, Twitter, and other web sites which results very little to the general objectives set out by most institutions.

It is therefore necessary for the network administrators to properly scrutinize bandwidth seeing that its utilization has grown exponentially. It can therefore be deduced, that except drastic measure are taken, the utilization of bandwidth will continue. There is, therefore, a need to mitigate this issue. In recent times, we have seen the organizations suffer a major setback due to improper use or management of bandwidth by its users. This practice has also eaten deep into the financial sector such that funds meant for the development of other facilities are channeled to the purchase of more bandwidth to meet up with the growing demands of its customers. As the usage of heavy bandwidth application grows and customers’ requests continue to rise; there is need for a conscious effort to screen, manipulate and control bandwidth utilization and proper implementation of effective, fine, and efficient bandwidth strategies.
In general, the proposed system is meant to aid institutions in allocating bandwidth resources to production services for the greatest demand.

Wairisal and Surantha (2018), proposed a design and evaluation of an efficient bandwidth management system for a corporate network based on the problem that usually occurs at every company which results to poor network performance such as low throughput and inefficient bandwidth utilization.

The design was performed using a top-down network design that considered the business and technical goals of the corporate entity in order to improve the bandwidth utilization efficiency. The system was evaluated using few traffic shaping methods in terms of bandwidth utilization, packet loss, and latency. The experiment results revealed that the proposed design using the per-connection queue (PCQ) traffic shaping method didn’t achieve the best performance.

Akpah, Mireku-Gyimah, and Aryeh (2017), proposed a system to improve the performance of large scaled computer network in the University of Mines and Technology (UMaT), Ghana, by managing the bandwidth utilization using squidGuard for the sole purpose of Internet facility to support teaching and learning, research and sharing of information. The researchers studied the behavior patterns and bandwidth utilization trends of the network users using Squid Analysis Report Generator in order to identify the causes of the challenges and effective deployment of bandwidth management control policies using squidGuard. The results revealed that the challenges of the network were attributable to misuse of the bandwidth mainly by some students on low-priority, bandwidth-hungry websites and applications such as pornographic and other useless websites and peer-to-peer applications; and lack of effective bandwidth management control policies.

Sanjeev, Vijay, and Sudarshan (2013) proposed a bandwidth allocation based on traffic load and interference in IEEE 802.16 mesh networks that improved the delay and throughput performance by proper utilization of assigned bandwidth. The proposed system is based jointly on traffic load and interference using a suitable path which is selected based on the least routing metric (RM) value. The simulated results revealed the high effectiveness of the proposed system which indicates higher bandwidth utilization and throughput as compared with existing nondiscriminatory end-to-end bandwidth allocation.

The current systems do not focus on the protocols which could expose further details on the activities of the users.

ii. The Propose System

The proposed system is an enhanced bandwidth control platform for effective monitoring and resource utilization in corporate networks that requires evolvement through gradual refinement from the initialization phase to the log capture phase. The functionality of the propose system is interpreted by the decomposition of the system into its components to improve it with a better procedure. The proposed system design is aimed at utilizing an extreme programming design process to ensure that our objectives are met by introducing some form of transparency to bandwidth usage, also ensuring that the users of the software have a correct and complete understanding of the software, and finally ensuring that the software is maintainable.

For proper understanding, a consideration of the design model which involves the data flow diagram, system architecture, and interfaces was considered based on the methodology adopted (Agile methodology), the data flow diagram is broken into modules as shown in Figure 1. In order to achieve a simple and easy to maintain software.

iii. Flowchart design

Logically, it processes the incoming data flow to outgoing data flow and helps store data in the repository of the system. The data flow notation is the pipelines through which the data flows and also included in the data flow chart are the codes known as the pseudocodes which are informal coding.
Figure 1 depicts the flow chart for variable declaration and initialization.
The flow chart in figure 2, shows how packets are captured and utilized. The Packet captures reads the packets by figuring out the length and character within the packets. From the length, it will determine if it is UDP or TCP. Then it starts to queries for the IP address and beta length, then pushes to the packets received and sockets.

There is a sub-procedure called packets to capture. It is connected to a particular interface that enables it to capture packets based on the variables defined or specified. The variables defined are the byte data, source, and destination of packets, IP address, and length of the packets. Thereafter these variables are transformed into what is known as strings. There is also a process list that causes multi-threading by starting and stopping the procedure. Once an interface is selected, the interface starts to listen to a packet that comes into

**Figure 2: Flowchart for capturing packet**
the interface. The interface starts to listen to packets by getting vital information as listed in the variables and converts it into strings and stores it into the data grid view. The process of doing this involves port listening, packets capturing and storing.

```vbnet
Dim procCMDWin As New Process
AddHandler procCMDWin.OutputDataReceived, AddressOf CMDOutput
procCMDWin.StartInfo.RedirectStandardOutput = True
procCMDWin.StartInfo.RedirectStandardInput = True
procCMDWin.StartInfo.CreateNoWindow = True
procCMDWin.StartInfo.UseShellExecute = False
procCMDWin.StartInfo.FileName = "cmd.exe"
procCMDWin.StartInfo.WorkingDirectory = "c:\"
```

Event listener: Console.keyPressed

- Has procCMDWin in exited?
  - Yes: procCMDWin.Dispose
  - No: procCMDWin.Start() procCMDWin.BeginOutputReadLine() swWriter = procCMDWin.StandardInput

Sub Routine: CMDOutput
```
Try
strResults = OutputLine.Data & Environment.NewLine
Invoke(uFin)
Catch ex As Exception
ex.ToString()
End Try
```

Sub Routine: UpdateText()
```
RichTextBox1.Text += strResults
RichTextBox1.SelectionStart = RichTextBox1.TextLength - 1
RichTextBox1.Focus()
intStop = RichTextBox1.SelectionStart
RichTextBox1.ScrollToCaret()
```
Figure 3, represents a flow chart for console prompt. It is basically designed to manipulate the software using a command prompt interface. This is achieved using a windows shell embedded inside Microsoft. There is a pro-C window which is a processing entity that enables the process to run in the background. It starts the redirection of standard input by hiding the windows interface of the shell to ensure it doesn’t display while running. It also converts the command prompt into the shell and executes it, then calls up the entity and brings out the standard output. Commands that can be executed include tracers, bandwidth, and this comes in a windowless form.

The sub-routing here is always there to listen to the strong keys events and this continued until the enter key is pressed, then it starts to compile and then execute and generates the standard output which in turn forwards it to the application to be displayed on the interface. The command output update takes the output generated and pushes it to the update test and the update test shows of the executed command are successful.
Subnet Scan

Dim myPing As Ping
Dim reply As PingReply
Dim addr As IPAddress
Dim host As IPHostEntry
Dim i As Integer = 1
ProgressBar2.Maximum = 254
ProgressBar2.Value = 0

i <= 255

Dim subnetn As String = "." & i.ToString()
myPing = New Ping()
reply = myPing.Send(subnet & subnetn, 900)

i = i + 1

If reply.Status = IPStatus.Success

Try
    addr = IPAddress.Parse(subnet & subnetn)
    host = Dns.GetHostEntry(addr)
    listVAddr.Items.Add({subnet, host.HostName, "Up"})
Catch
    MessageBox.Show("Couldnt retrieve hostname")

cmdScan.Enabled = True
cmdStop.Enabled = False
txtIP.Enabled = True
lblStatus.Text =

MessageBox.Show("Scanning done!" "Found " count.ToString() " hosts.")

Stop
Figure 4 shows the flowchart of the subnet scan. Here, certain variables are defined which are used to collect information such as the ping, address, reply host, address number of hubs and iteration. Since each network address point is from 255 to 0. The initial value of the progress scan is zero and the maximum value of the progress bar is 255. This is done such that when the net scan starts, it starts counting from one up. When the scan starts, it uses a counter to determine the number of cycle completed. It checks to see if the first number of cycle is up to 255, if it says NO, it will take that network address and add 1 as the host address and push it. Then it pings that particular IP address, if it response, then it record it and get information about that particular IP address, the operating system, the host name, open ports. If it does get all these information stated above then it will return an error and push it to the grid view where it is displayed. But if it is alive, it generates all the information stated. Once it returns this IP address alive, it is passed to the subnet sub-routing to get the host using the get-host entry address by specifying the host that are alive and push it to be display. This iteration is between 1 and 255, depending on specification. It has two option to start and stop the scan which is the scan start and scan stop and once it is done, the status of the scan is displayed on the interface and the counter tells us if the scan is completed.
**Figure 5: Block diagram of proposed Network Monitoring System**
In Figure 5 the block diagram has eleven processes or phases i.e. starting from Process 0 – 11, with different functional activities being carried out in order to actualize our desired objectives as the case may be.

**The process 0** starts with the initialization of the system; this phase is enabled through Microsoft visual studio which contains tools such Software Development Kits (SDK) and pre-requisites for the functionality achieved during this phase.

There is status trip tool which is used to display the overall status of the application. During the initialization process, there is a self-test carried out on the interface of the system to ascertain the health of the system before lunching.

**The Process 1 and 2** phases is the login phase where the users’ credentials are verified by some authentication process. If a user is certified ok, then access right is granted. The users can also be properly managed.

**The Process 3** is a dashboard browser-based integration that helps display the functionality and capability of the proposed system. Here, the Pandora SDK is integrated into the interface of the dashboard to display a brief overview of the extended functionality of the software-based system. The dashboard has the .Net Framework browser integrated directly into it. This browser on like the web browser tool found in the Visual Studio toolbox has the capability of handling script and routine calls made to and fro the server as the case might be. The data which is displayed on this dashboard is processed at the backend of the software by the server.

**The Process 4** is the scanning phase. The subnet Scanner is one of the most important interfaces in proposed network monitoring system. This interface is used for the scanning of every node on the network. This is done by specifying the network address for the subnet which needs to be scanned; when the scan is initiated, the software gets the status of each node (alive or dead), using a simple ping test which is done twice before moving to ping the next node. If the node is alive, it goes ahead to query it for the OS type, Device name, Username, Machine type, etc. This information is then stored in the grid view object within the interface. This information is not stored permanently within the software. The scan goes on 24 hours at the background but can be stopped as well as resumed at any time the administrator wishes to do so. It also contains the status label which shows the current status of the overall scan whether active or passive. This interface also allows for the allocation of bandwidth size to any device on the network using the bandwidth allocation tool which was built into the software. Using administrative privileges over the network and depending on how the network is configured, certain tasks which include shutting the host remotely, disabling the port where the device connected to can be done remotely from this interface. The main purpose for network scanning is as follows

(a) Recognize available UDP and TCP network services running on the targeted hosts.
(b) Recognize filtering systems between the user and the targeted hosts.
(c) Determine the operating systems (Oss) in use by assessing IP responses.
(d) Evaluate the target host’s TCP sequence number predictability to determine sequence prediction attack and TCP spoofing.

**The Process 5** is the protocol separator. This is the phase where we separate the protocols into UDP and TCP for proper traffic analysis so as to determine traffics that gain access to the network.

**The Process 6 and 7** is the packets capture and analyser phase. This module is designed for intercepting any data packet that is crossing or moving over a specific computer network. Once a packet is captured, it is stored temporarily so that it can be analyzed. These packets are reassembled once all the data packets reach their intended destination. This interface acts as a packet sniffer on a network; the sniffer intercepts the network traffic and captures the raw data packets. Data can be gotten from one active network interface of a device at a time, depending on which network interface card is selected before the packet capture.
The Process 8 is the control stage. In this stage we list out sites we want to gain access into our network. Once a user tries to access a site that was not listed, the user is denied access to that site. In the event that the user uses a tor to access the site, we have also put in place special mechanism to bar such sites or applications from accessing the sites.

The Process 9 is the firewall manager. The Firewall Manager is based on IP tables with a progressive load design that includes and supports all the regular management options of a firewall. Moreover, it has support for load balancing, label-oriented roll management, change log, and easy support to create rules out of recorded blocked traffic. This is all based on the type of firewall application running within the network environment which can be managed from within the proposed system firewall management interface.

The Process 10 is the packet detail pane shows the current packet (Selected in the “Packet List Pane”) in a more detailed form. This pane shows the protocols and protocol fields of the packet selected in the “Packet List” pane. It also shows the source and destination IP address of the captures packet fragments.

The Process List determines the actual processes or lists that currently run on the system. You can use the TaskList command to display a list of currently-running tasks. Tasklist displays the process ID number for each running task, the name of the executable program that started the task, and, when available, the window title. You can also optionally display the process priority, the modules (dll’s) loaded by that process. The process list uses .Net Browser provides .NET WPF / WinForms Control that can be embedded into .NET Applications to display HTML content including modern web pages. With Load Handler one can handle any load activity including filtering all URLs loaded in Browser component.

iv. Conclusion

The primary findings of this study is to provide a simplified and transparent solution or method to network/system administrator in addressing some of the challenges faced by insurance institutions in the proper management of its bandwidth. This system will help network administrator respond quickly, easily and timely to potential issues that may arise during activities periods

Acknowledgment

I would like to express my special appreciation to my supervisor and the Head of Department Information Technology Federal University of Technology Owerri; Engr. Dr. Obi C. Nwokonkwo for the major supervisory role and encouragement throughout this work.

References


AUTHORS

First Author – Engr. Chimezie Oji, B.Eng, Electronics Engineer, MNSE, MCP, ITIL, MCSA, Chimexmr@gmail.com

Second Author – Engr. Dr. Obi Nwokonkwo, B.Eng, MSc, PhD, Obinwokonkwo@gmail.com

Third Author – Dr. John-Obum, MNCS, CPN, Reg, MIEEE, IAENG, adetokunbo.johnotumu@futo.edu.ng

Fourth Author – Ms. Adu Folashade, B.Sc Computer Science; PGD Information Management Technology, xtiana1202@gmail.com