DOCUMENTATION

LUMIERESOMBRE SERVER – ADMINISTRATOR REMOTE CONTROL TOOLS

The following document describes the process of building a tool as the user (i.e. client) requested that would answer to all the system and feature requirements set forth.

Project development started in the 05/2020 as a result of a contractual agreement between the *LumiereSombre@Team* and the client. In the following pages the process of development is thoroughly described and put to test by the minimum acceptance criteria mutually agreed upon with the sole condition of providing regular updates until all the criteria/features are met.

Project Name must be **Lumiere Sombre Server.**

LumiereSombre@Team

Contact name Reznicencu Bogdan

Team rep. reznicencu.bogdan99@gmail.com

Repo. Git. https://github.com/ReznicencuBogdan
Telephone no. +40 (744 368 823)

''' This project is a continuation of an old project of mine that started in 2013. It was part of a
''' local competition that involved 'state of the art'. CNC devices and means of communications.

Index

User story

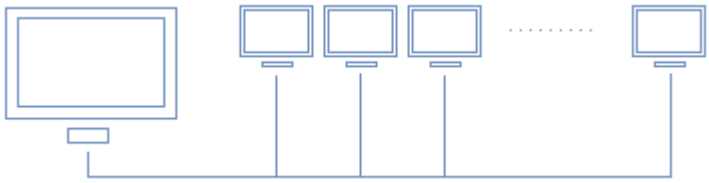


Fig 1.1 Administrator device controlling

organization's devices

EPIC	USER STORY	ACCEPTANCE CRITERIA
As a System Administrator, I need to access all the computers discovered in the organization's range with ease from a secure platform so that I can manage the everyday issues and system update requirements.	As a System Administrator, I need to be able to login into the platform so as to perform specific tasks on specific clients.	• Log into platform • Select specific client • Trigger events on behalf of the client
	As a System Administrator, I need to be able to easily configure the server.	
	As a System Administrator, I need to be able to identify specific clients and tag them for easier later use.	
As a System Administrator, I need to select one specific client and remotely manipulate the data/system resources without requiring any sort of user interaction on the client's end.	As a System Administrator, I need to be able to interact with the filesystem I.e. read, write, execute -- rwxe on the behalf of the client.	• Filesystem access – Create new file – Create new folder – Rename file – Rename folder – Delete file – Delete folder

	As a System Administrator, I need to be able to manipulate the registry data hive.	– Move file – Move folder – Enumerate folders – Enumerate files – Modify file content – Upload file – Upload folder – Download file – Download folder • Registry hive access – Create key – Create subkey – Enumerate keys – Enumerate subkeys – Rename key – Rename subkey – Delete key – Delete subkey – Modify subkey value • Registry hive access – Run shell batch script – Run powershell scripts – Run PE/executable files – RunAs
	As a System Administrator, I need to be able to issue shell/batch/powershell scripts on behalf of the client.	
As a System Administrator, I need to be able to physically manipulate the client.	As a System Administrator, I need to be able to observe the screen activity of the client.	• Screen Video stream. • Manipulate cursor
As a System Administrator, I need to be able to be able to restrict specific clients.	As a System Administrator, I need to be able to be able to ban client by unique id or by IP (worst case scenario).	• Ban user based at least based on IP.

As a System Administrator, I need to be able to be able to induce a BSOD ((funny memories - copied from the first version developed in 2013)) action on behalf of a selected client. The server will send a command to the client telling him that the client should induce a Forced Shutdown. This should be useful in case of an attack in which case forceful shutdown is usually suggested so that the percentage of data corruption is minimum. Sometimes this idea is not that great as some of the most known encryption-malware have either weak encryption – and therefore the keys may still reside in the ram of the machine or the keys could be cracked and the files restored. Also forcing a shutdown may leave a file in the process of encryption in an unstable state and therefore become irrecuperable.

Minimum System Requirements	
Server must be compatible with the following systems	Windows XP SP3,1 Windows Vista SP1, x86 32-64 Windows 7 SP1-4, x86 32-64 Windows 8 , x86 32-64 Windows 8.1, x86 32-64 Windows 10 1607, x86 32-64 Windows 10 1790, x86 32-64 Windows 10 1903, x86 32-64 Windows 10 1909, x86 32-64 Windows 10 2004, x86 64
Server must be able to run on the following system configurations	RAM: Minimum 256MB CPU: Any I-x86, AMD HDD: Minimum 40GB
Server must be able to run on the following network configurations	Gigabyte, Realtek, any eth,wlan interfaces.

Product efficiency requirements

- The server must be fast and easy to use.
- Must provide intuitive design and snappy transitions from one stage to another
- Must provide compression algorithms so as to use a low network signature
- Must provide efficient parallel UI triggered events.

Product impact on physical resources
At any time one client activity can be transferred to background and other be brought back to focus I.e. the downloading/uploading operation. At any point in time, the network usage signature must be low (I.e. under ~20mb/s). The LZ4 compression algorithm must be used especially when it comes to video stream transfer. At any point in time the CPU signature usage shouldn't exceed half of the max CPU freq.

Build Tools
• The project must be developed by using the IDE developed by Microsoft, Visual Studio 2019 Enterprise with License, v19.6 according to the code standards. • The code must be written in the .NET Framework family language in Visual Basic, targeting a version of Net Framework of 4.6 or higher. • Lower versions of .NET Framework can be used if absolutely needed. • The code must be able to run independently of the development environment and must be able to run without needing direct access to the source code and the compilation tools.

Coding Standards, Style

The entire server-project must be developed in a managed environment I.e. .NET Framework.

Visual Basic is the language of option in which the server must be created.

Functions and methods must be developed by following the K&V's codding style, I.e. by using well commented and logically named functions.

Example:

```
''' <summary>

'''  Initialize the drawing library before hand

''' </summary>
```

```
''' <param name=" argument1"></param>

''' <returns></returns>

public sub InitializeDirectUIComponents(ByVal argument1, …., etc.)
```

External Libraries

The use of external libraries whether they be imported via nugget transaction system or manually is allowed as long as the licensing terms are subjected to the MIT/The Unlicense license.

Examples of libraries the server should be comprised of

- LZ4 – MIT Licensed Library used mostly for compressing loosely large amounts of data in managed/unmanaged data streams.
- MetroModernUI.1.4.0.0 - MIT Licensed Library that offers an UI overlay on top of the standard windows forms.
- Be. Windows.Forms.HexBox.1.6.1 - MIT Licensed Library that offers fast and reliable hex editor.

Networking

The networking implementation must be based on the private ArchTCP simplex protocol that would provide one-way multichannel communication possibilities with each client individually.

Pinging system must be independently developed as it is excluded from the IEEE simplex protocol. This is a must be implemented so as to easily identify non active client or undefined socket timed out behavior.

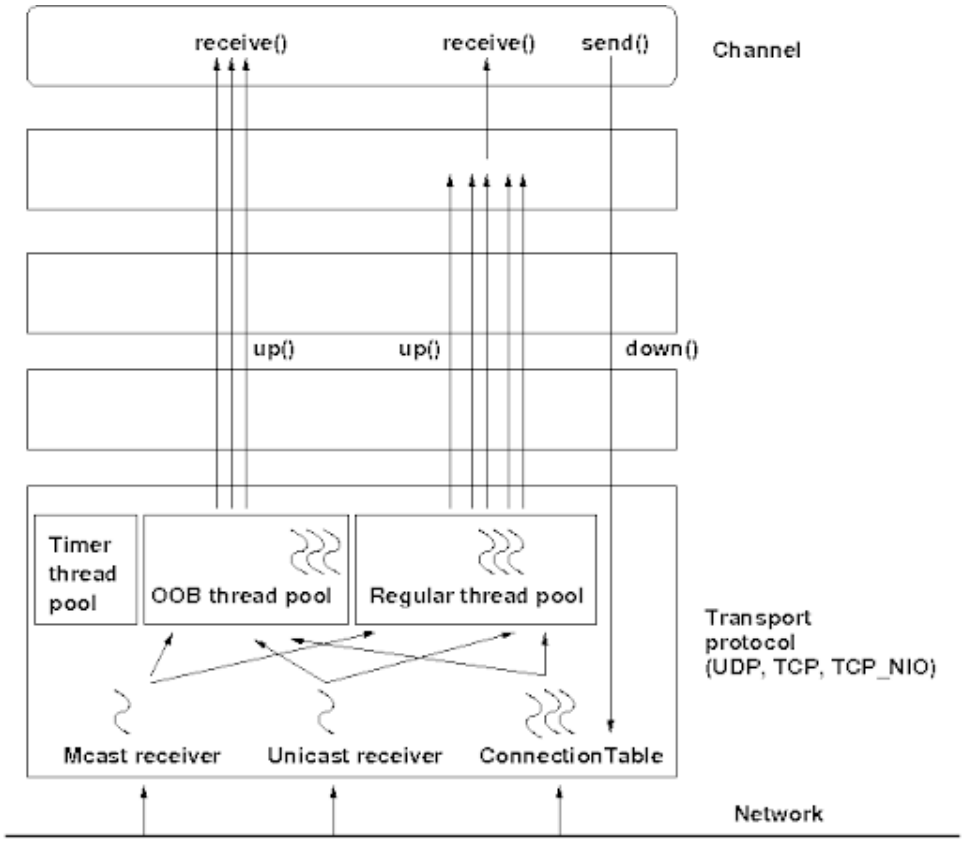


Fig 1.2 A diagram of one version of the simplex algorithm.

Both the socket connection timeout as well as connection timeout must be handled in the server:

- A connection timeout occurs only upon starting the TCP connection. This usually happens if the remote machine does not answer. This means that the server has been shut down, you used the wrong IP/DNS name, wrong port or the network connection to the server is down.

- A socket timeout is dedicated to monitor the continuous incoming data flow. If the data flow is interrupted for the specified timeout the connection is regarded as stalled/broken. Of course, this only works with connections where data is received all the time.

Server Network Handshake	
	1. Server\Client will push the stream into SSL\TLS 2. Client connects to server

Diagram of server client connection handshake

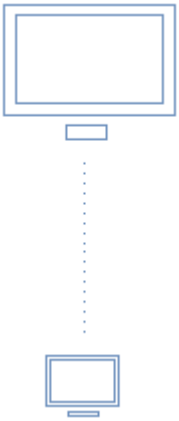


Fig 1.3

3. Server sends handshake request
4. Client receives handshake request along registration token.
5. Client salts token with unique id.
6. Send handshake acknowledge along with a private key
7. Server verifies new token, registers key, sends new key
8. Client receives new key and sets it as default token.

The protocol works as follows. The server can receive a response from the client only as long as the responses token is still registered inside the server. If the token expires or is a duplicate of another pending token then the response can be considered as being part of an echoing attack and the connection will be dropped and the client banned.

If the token is just expired then the server will send a expire packet as to notify the client that any token still registered identical to the one issued by the server is to be expired and re-registered.

Once a client is successfully connected to the server, the server will request for a unique identification structure that will identify uniquely the client.

The server will map the client instance to the newly received structure in memory.
Each registered action on the server side will be associated to the currently selected client in a list if any and will be sent to that particular client.
Multiple client multi action request is not supported by the protocol.
Banning a user will result in associating that specific's user unique identification key to a banned token flag that will for the entire existence of that specific flag trigger connection forced shutdowns. This will therefore prevent unauthorized client access to the server's communication protocol as it was deemed untrustworthy.

Each received packet will be thorughly checked by the server.
There will be two mechanisms of prevention installed

1. The mechanism that check for a specific magic checksum installed in the beginning of the socket packet that was received.
   The magic checksum will be agreed to be some unsigned int.

2. A mechanism that checks for the integrity of the packet. Even if the TCP and SSL/TLS technologies do also provide some reliable way of checking for misalignment or packet errors on the way, another such protection measure will be installed in order to prevent tragic errors of communication or of attacks in case the private keys are discovered. Therefore, the following algorithm should be installed

```
Dim mComputedChecksum As UInt64 = 0
mComputedChecksum += ccmPacket.packetSize
mComputedChecksum = mComputedChecksum Xor &H7F12A071
```

	mComputedChecksum += ccmPacket.dataType mComputedChecksum = mComputedChecksum Xor &H7F12A071 Dim nullChar As Integer = ccmPacket.arguments.IndexOf(vbNullChar) Dim size As Integer = IIf(nullChar = -1 Or nullChar = 0, ccmPacket.arguments.Length - 1, nullChar - 1) For argItterator As Int32 = 0 To size Step 1 mComputedChecksum += Convert.ToByte(ccmPacket.arguments(argItterator)) mComputedChecksum = mComputedChecksum Xor &H7F12A071 Next Return mComputedChecksum The following structure is to be taken into consideration: Dim magicStart As UInt64 ' 0-7 Dim packetSize As UInt64 ' 8-15 Dim dataType As UInt32 ' 16-19 Dim arguments As String ' 20-1043 Dim checksum As UInt64 ' 1044-1051 Dim token As String ' 1052-1307 By default, the protocol supports packet magic checksum checking in order to provide some sort of protection against misalignments. Therefore, magicStart holds 0xFFFFFFFFFFFFFFFF PacketSize holds 0xFFFFFFFFFFFFFFFF DataType holds 0xFFFFFFFF Checksum holds 0xFFFFFFFFFFFFFFFF More details with regard to the networking can be found on the https://github.com/ReznicencuBogdan/ArchTCP Or https://github.com/LumiereSombre/ArchTCP The protocol is well defined and the

How the software is supposed to work

Features

The process is straightforward. The administrator double clicks on the server application installed locally if it is installed or builds the solution with the tools the solution was built with.

The Server starts and a simple and efficiently positioned interface is shown. From the interface the administrator I.e. the user will be able to select the actions he wants to take.

The interface will provide the following options on the left-hand side of the application:

- Client identification tools:
    a. Unique Identification – String of 20 bytes that will uniquely identify a client in the list of active clients connected to the server.
    b. Username – another identification string that will identify the client by the name of the computer connected.
    c. Custom Alias – as one can't change the Username of a client on a Windows Platform (I.e I, the Administrator don't want to change the Username of that client) the custom alias will be used for faster association between a name and a frequently used computer.
    d. Extra data – another string that will hold a large string of data – it can hold any kind of information related to that specific client.
    e. Pc Description
    f. Os Version – this datatype will hold the type of the architecture the client is running on I.e. if it is 32-bit version will show 32 otherwise will show 64bit.
    g. Version – an integer that will hold the version of the client running on the machine connected to the server.

- Next, the interface will provide a list of buttons that the user I.e. the administrator will be able to click. The buttons will be named in the following order:
    a. "Terminal status – force "- on click the server will send a command to the client telling him the terminal component should be forcefully opened.
    b. "Ftp – force "- on click the server will send a command to the client telling him the ftp component must be forcefully opened.
    c. "Upload Manager – force "- on click the server will send a command to the client telling him the upload manager component must be forcefully opened.
    d. "Download manager – force "- on click the server will send a command to the client telling him the Download Manager component must be forcefully opened.
    e. "Regedit – force " - on click the server will send a command to the client telling him the Regedit component must be forcefully opened.
    f. "Video Stream – force "- on click the server will send a command to the client telling him the video stream component must be forcefully opened.

- Next the interface will provide the following list of buttons:

a. “Live Stream” - on click the server will send a command to the client telling him that the video streaming process should start.
b. “Camera” - on click the server will send a command to the client telling him that the camera streaming process should start.
c. “BSOD” - (funny memories - copied from the first version developed in 2013). on click the server will send a command to the client telling him that the client should induce a Forced Shutdown. This should be useful in case of an attack in which case forceful shutdown is usually suggested so that the percentage of data corruption is minimum. Sometimes this idea is not that great as some of the most known encryption-malware have either weak encryption – and therefore the keys may still reside in the ram of the machine or the keys could be cracked and the files restored. Also forcing a shutdown may leave a file in the process of encryption in an unstable state and therefore become irrecuperable.
d. “View local files” - in which case the local folder where the downloaded client’s resources reside in, is opened in the explorer.
e. “Ban” - on click the server will insert the currently selected client and insert it into the database of blacklisted connections. Afterwards the connection to the client is forcibly destroyed.
f. “Destroy” - on click the server will destroy the connection to the currently selected client.

- Next the interface will provide a text component in which critical messages will be printed on so that the user I.e. the administrator will be able to tell what errors are occurring in the background during the exchange of transactions.

- Next the interface will show an ftp component that will allow access to the client’s filesystem. The component will provide the following actions:
    a. Enumerate folders or files in the current directory
    b. Enter directory
    c. Create directory
    d. Create file
    e. Rename folder
    f. Rename file
    g. Delete folder
    h. Delete file
    i. Run file
    j. Download Folder
    k. Download File
    l. Edit File

The ftp component will also provide an URL-bar for faster access and also the usual filesystem shortcuts – to the Desktop, AppData, Roaming,Temp,Windows, System32, Startup, My Computer.

More than that, with regard to the Download Manager and Upload Manager the interface will also provide two more buttons:

1. Cancel Download – on click the current download transaction will be aborted.
2. Cancel Upload – on click the current upload transaction will be aborted.

- Next the interface will provide an input text component that will act like a terminal that will accept all the batch commands. The component will work as follows:
  - The user will provide a command and on [ENTER] click the command will be sent to the client and the shell component will provide the response afterwards in the same component on the next line.
  - The component will also provide a fast typing method by means of command history that will be accessible through the [UP], [DOWN] buttons. The history will also be saved in another visual component that will stack the list of commands in order.

- Next the interface will provide a mirror to regedit on the client's machine. The component will have the form of a TreeView and by default will show the following keys in order:
  - "HKEY_CLASSES_ROOT"
  - "HKEY_CURRENT_USER"
  - "HKEY_LOCAL_MACHINE"
  - "HKEY_USERS"
  - "HKEY_CURRENT_CONFIG"

After clicking any of the nodes provided before, the respective node will be expanded and the client will provide the key-list content of that node, and will populate the values view.

The component will be comprised of two views:

i. The keys view
ii. The values view

On clicking any of the subkeys/keys node a context menu will be shown which will provide the following items:

i. Refresh – will update the subkeys of the selected node.
ii. Delete – will delete the selected node., New Key – will create a new subkey, son to the selected node.

On clicking on the values component, on an empty space the following menu will be shown:

i. Create new Value: On click a new Window will be shown in which the following will be shown:
    - A textbox in which the name of the value will be entered
    - A box in which the value type will be entered: “REG_SZ”,” REG_BINARY”, “REG_DWORD”, “REG_QWORD”, “REG_MULTI_SZ”, “REG_EXPAND_SZ”
    - A hexeditor in which the data of the value will be inserted in hexadecimal.

i. Refresh – the list of values will be refreshed.

On clicking on the values component, on an existing value the following menu will be shown:

i. Rename Value
ii. Delete Value

On double clicking a value, the editing window will be shown which is exactly the same as the window previously described.

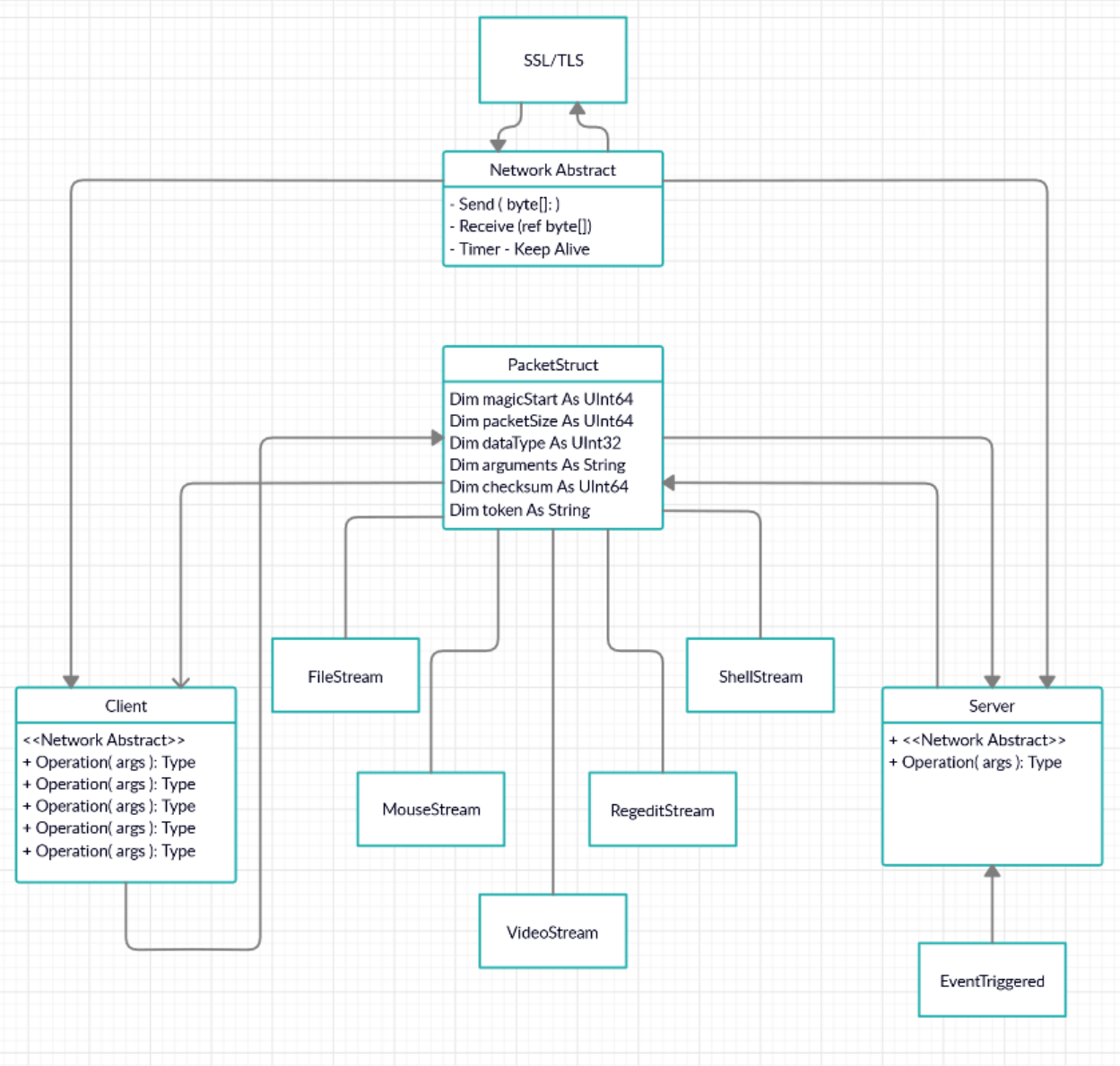
- Next, the interface will provide a menu/button by which a window that manages the user database will be opened.

  The window will provide a GridView in which all the registered users will be shown in real time, and another GridView in which all the blacklisted clients will be shown.

  Also a button will be shown that will be used for moving a registered user to the blacklist and back.

Note: For a reliable experience the Be.Hex.Editor library mentioned in the previous article is to be used.

UML Design of The Server



A brief explanation of the protocol ArchTcp. The protocol is not part of the project as it is a standardized protocol and doesn't therefore make the purpose of the tool. More details can be found on the following site https://www.ieee-archt4.org/open-sourc/~doc

The way the server communicates with the client is by sending data by first providing a stream of bytes also known as a header that will arrive in order on the client side. The packet describes the data that is to be received. Data includes and isn't limited to, the size of the payload, the amount of shifting required since last packet, the padding required for faster unmanaged byte processing, the validation toolkit I.e. the token which includes a validation salt, checksum and timestamp – needed to tell whether a client is delayed by some intensive task, or worse whether the packet received is part of a replication attack.

The client will respond in the same manner, the only difference this time consisting in the fact that the token will not be regenerated but rather forwarded after having been modified with some flags.

The reason why the server and the client are able to communicate in a multithreaded way with a high-level of packet loss in cases such as the following: (

a. Downloading an entire folder of GBs, while,
b. Uploading an entire folder of GBs, while
c. Live Streaming the Desktop, while
d. Modifying the Regedit, while
e. Receiving large amounts of shell query commands

), is that the client by itself doesn't really exist.

As a matter of fact, there is no single client installed on the remote machine, but rather there is a collection of clients installed on the remote machine that communicate with each other on a safe and multithreaded way. They individually connect to the server, offering of course the same unique identifiers as each and every single one of the clients serve on the same machine.

As a result of such of an approach, the ArchTcp protocol decides that each connected client that possess the same unique identifier as another client in the list of connected clients is actually part of a higher hierarchy or collection that builds up the client.

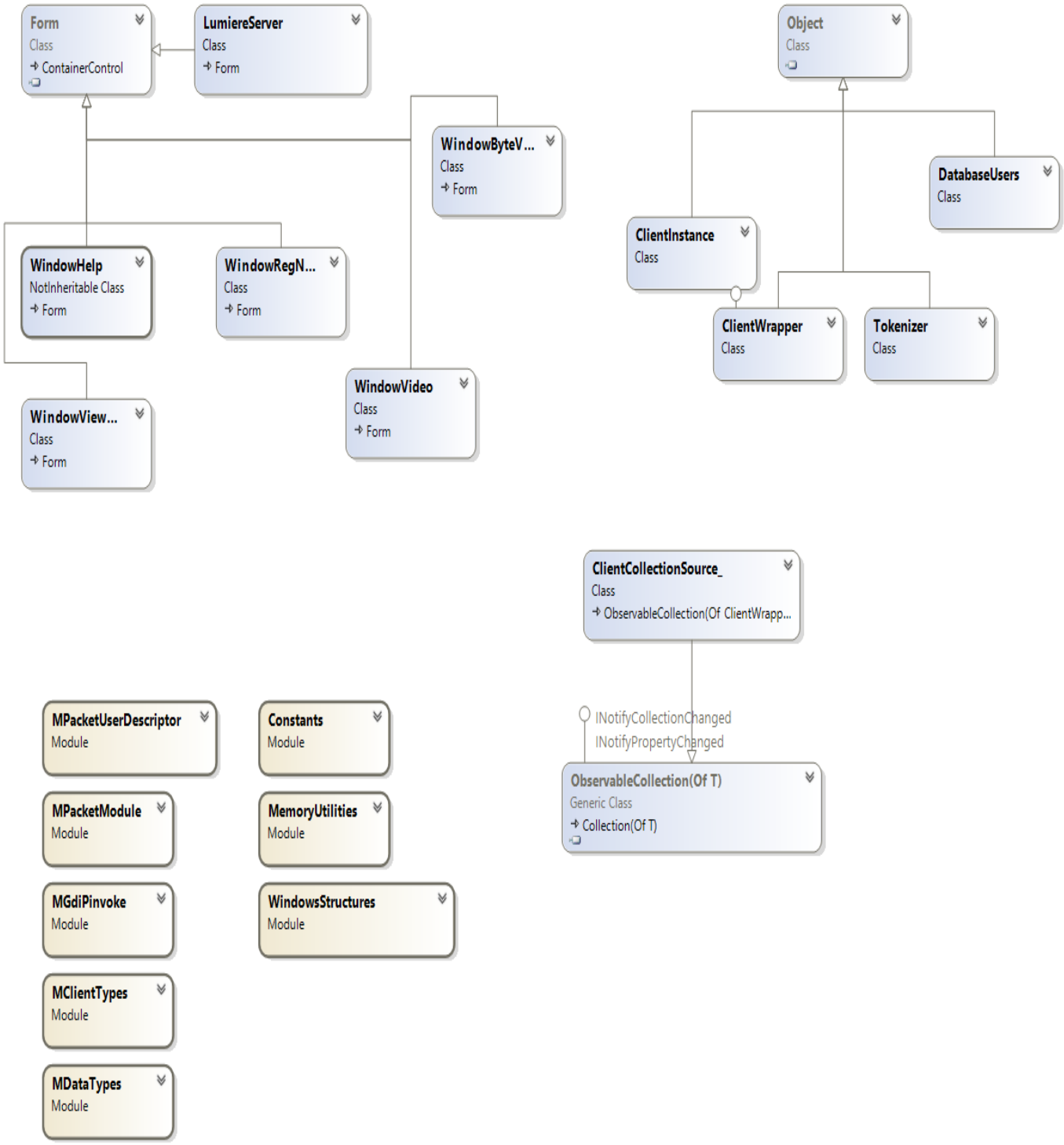
Therefore, when a 'client' is selected from the list of connected clients so as to be able to send commands to that client, one is actually selecting a collection of clients. The server decides based on the architecture of the command to what kind of a client that specific command should be sent to.

On the server, a collection wrapper, or the collection that holds the clients of the same machine, builds an exhaustive worker pool of threads used for each 'component' installed on the client machine. As a result of this type of architecture, the server is able only to decide whether a client not selected is active or not and it is limited to that. One unselected client can't be active and receive responses if it wasn't selected and issued a command. One unselected client can only be removed from the list by the server only if the low-level check of universal socket timeout which is issued by the server proves that the socket is no longer valid, and that's all.

So, this answers the following technical question: what happens if the client's socket buffer accumulates too much data due to retransmission of packets not read by the server? The protocol doesn't allow this.

Another technical question would be: what if the server's internal socket buffer accumulates too much data due to the open connection? Again, the protocol doesn't allow such a conduit on the client's side. But what if the client is malformed? The ArchTcp protocol states that during the low-level check of universal socket status, the server should use fd_read so as to check whether there is any data in the sockets buffer. If there is but the client associated to the connection is not selected or it states that there is no need for packet shifting (there was no shift of data during last transmissions) then that client collection must be dropped. QED.

UML Diagram



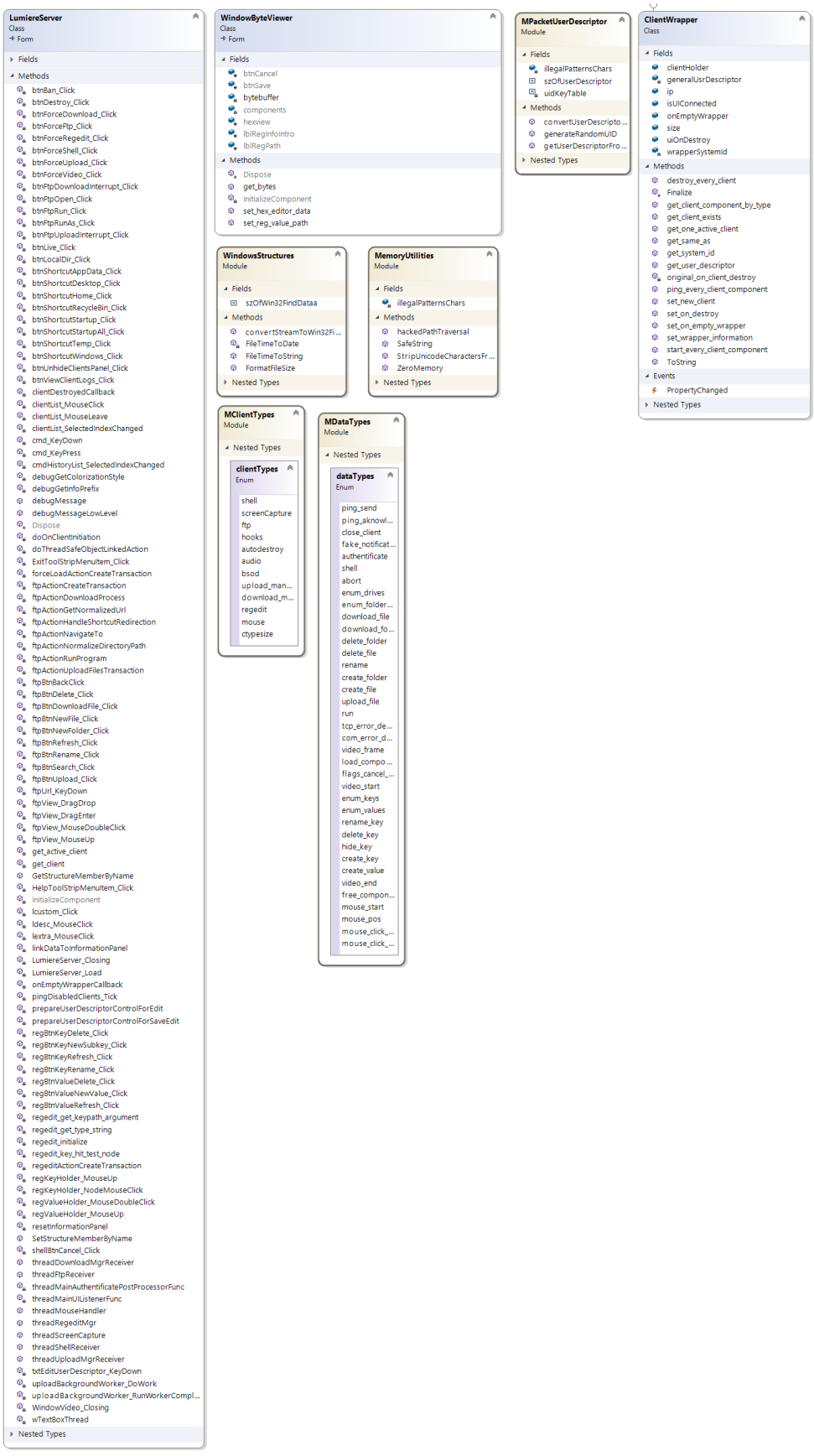
The diagram above is a suggestion of how the classes should be structured like.

Note the following, in the above diagram:

a. The Class Form is a base for the following forms:
    i. WindowHelp
    ii. LumiereSombre – main Window
    iii. Window Registry

b. The ClientCollectionSource is implementing the ObservableCollection for real time events on collection modifications.

Class Diagram Of Intended Classed


LumiereServer
Class
Form
Fields
Methods
btnBan_Click
btnDestroy_Click
btnForceDownload_Click
btnForceFtp_Click
btnForceRegedit_Click
btnForceShell_Click
btnForceUpload_Click
btnForceVideo_Click
btnFtpDownloadInterrupt_Click
btnFtpOpen_Click
btnFtpRun_Click
btnFtpRunAs_Click
btnFtpUploadInterrupt_Click
btnLive_Click
btnLocalDir_Click
btnShortcutAppData_Click
btnShortcutDesktop_Click
btnShortcutHome_Click
btnShortcutRecycleBin_Click
btnShortcutStartup_Click
btnShortcutStartupAll_Click
btnShortcutTemp_Click
btnShortcutWindows_Click
btnUnhideClientsPanel_Click
btnViewClientLogs_Click
clientDestroyedCallback
clientList_MouseClick
clientList_MouseLeave
clientList_SelectedIndexChanged
cmd_KeyDown
cmd_KeyPress
cmdHistoryList_SelectedIndexChanged
debugGetColorizationStyle
debugGetInfoPrefix
debugMessage
debugMessageLowLevel
Dispose
doOnClientInitiation
doThreadSafeObjectLinkedAction
ExitToolStripMenuItem_Click
forceLoadActionCreateTransaction
ftpActionCreateTransaction
ftpActionDownloadProcess
ftpActionGetNormalizedUrl
ftpActionHandleShortcutRedirection
ftpActionNavigateTo
ftpActionNormalizeDirectoryPath
ftpActionRunProgram
ftpActionUploadFilesTransaction
ftpBtnBackClick
ftpBtnDelete_Click
ftpBtnDownloadFile_Click
ftpBtnNewFile_Click
ftpBtnNewFolder_Click
ftpBtnRefresh_Click
ftpBtnRename_Click
ftpBtnSearch_Click
ftpBtnUpload_Click
ftpUrl_KeyDown
ftpView_DragDrop
ftpView_DragEnter
ftpView_MouseDoubleClick
ftpView_MouseUp
get_active_client
get_client
GetStructureMemberByName
HelpToolStripMenuItem_Click
InitializeComponent
lcustom_Click
ldesc_MouseClick
lextra_MouseClick
linkDataToInformationPanel
LumiereServer_Closing
LumiereServer_Load
onEmptyWrapperCallback
pingDisabledClients_Tick
prepareUserDescriptorControlForEdit
prepareUserDescriptorControlForSaveEdit
regBtnKeyDelete_Click
regBtnKeyNewSubkey_Click
regBtnKeyRefresh_Click
regBtnKeyRename_Click
regBtnValueDelete_Click
regBtnValueNewValue_Click
regBtnValueRefresh_Click
regedit_get_keypath_argument
regedit_get_type_string
regedit_initialize
regedit_key_hit_test_node
regeditActionCreateTransaction
regKeyHolder_MouseUp
regKeyHolder_NodeMouseClick
regValueHolder_MouseDoubleClick
regValueHolder_MouseUp
resetinformationPanel
SetStructureMemberByName
shellBtnCancel_Click
threadDownloadMgrReceiver
threadRtpReceiver
threadMainAuthentificatePostProcessorFunc
threadMainUIListenerFunc
threadMouseHandler
threadRegeditMgr
threadScreenCapture
threadShellReceiver
threadUploadMgrReceiver
txtEditUserDescriptor_KeyDown
uploadBackgroundWorker_DoWork
uploadBackgroundWorker_RunWorkerCompl...
WindowVideo_Closing
wTextBoxThread
Nested Types
WindowByteViewer
Class
Form
Fields
btnCancel
btnSave
bytebuffer
components
hexview
lblRegInfoIntro
lblRegPath
Methods
Dispose
get_bytes
InitializeComponent
set_hex_editor_data
set_reg_value_path
MPacketUserDescriptor
Module
Fields
illegalPatternsChars
szOfUserDescriptor
uidKeyTable
Methods
convertUserDescripto ...
generateRandomUID
getUserDescriptorFro ...
Nested Types
ClientWrapper
Class
Fields
clientHolder
generalUsrDescriptor
ip
isUIConnected
onEmptyWrapper
size
uiOnDestroy
wrapperSystemId
Methods
destroy_every_client
Finalize
get_client_component_by_type
get_client_exists
get_one_active_client
get_same_as
get_system_id
get_user_descriptor
original_on_client_destroy
ping_every_client_component
set_new_client
set_on_destroy
set_on_empty_wrapper
set_wrapper_information
start_every_client_component
ToString
Events
PropertyChanged
Nested Types
WindowsStructures
Module
Fields
szOfWin32FindDataa
Methods
convertStreamToWin32Fi...
FileTimeToDate
FileTimeToString
FormatFileSize
Nested Types
MemoryUtilities
Module
Fields
illegalPatternsChars
Methods
hackedPathTraversal
SafeString
StripUnicodeCharactersFr...
ZeroMemory
Nested Types
MClientTypes
Module
Nested Types
clientTypes
Enum
shell
screenCapture
ftp
hooks
autodestroy
audio
bsod
upload_man...
download_m...
regedit
mouse
ctypesize
MDataTypes
Module
Nested Types
dataTypes
Enum
ping_send
ping_aknowl...
close_client
fake_notificat...
authentificate
shell
abort
enum_drives
enum_folder...
download_file
download_fo...
delete_folder
delete_file
rename
create_folder
create_file
upload_file
run
tcp_error_de...
com_error_d...
video_frame
load_compo ...
flags_cancel_...
video_start
enum_keys
enum_values
rename_key
delete_key
hide_key
create_key
create_value
video_end
free_compon...
mouse_start
mouse_pos
mouse_click_...
mouse_click_...

BACKLOG

List of implemented features

The following features were implemented during the development procedure of the LumiereSombre Server Software. If it is written “To-do” in front of the feature then that feature is still in the development phase.

- To-do: Log into platform
- Done: Select specific client
- Done: Trigger events on behalf of the client

- Filesystem access
  - Done: Create new file
  - Done: Create new folder
  - Done: Rename file
  - Done: Rename folder
  - Done: Delete file
  - Done: Delete folder
  - Done: Move file
  - Done: Move folder
  - Done: Enumerate folders
  - Done: Enumerate files
  - Done: Modify file content
  - Done: Upload file
  - Done: Upload folder
  - Done: Download file
  - Done: Download folder

- Registry hive access
  - Done: Create key
  - Done: Create subkey
  - Done: Enumerate keys
  - Done: Enumerate subkeys
  - Done: Rename key
  - Done: Rename subkey
  - Done: Delete key
  - Done: Delete subkey
  - Done: Modify subkey value

- Registry hive access
  - Done: Run shell batch script
  - Done: Run PowerShell scripts
  - Done: Run PE/executable files
  - Done: RunAs

- Done: Screen Video stream.
- Done: Manipulate cursor
- To-do: Video Camera Stream

- Done: Ban user based at least based on IP.
- Done: Induce a forced BSOD (funny memories - copied from the first version developed in 2013).

Testing Protocols

Testing the protocols for improved experience

Create a NUniteTest solution in .NET Framework

1. Usability test
   a. Testing the UI interface for bugs or unintended behavior under different circumstances which involve creating a virtual environment running tens of different collections of clients. This tests can reveal undesired concurrency issues/race conditions if some modifications are made to the code.

      NOTE: This test is not available because that would suppose that the client source is available to the developer.

   b. Testing the positions of the buttons under different resolutions to check for any undesired positional displacements. Manual check
   c. Testing the database functions from LumiereServer.ClientInstance().
2. Connection Timing Test
   a. Testing the networking protocol for uncaught socket exceptions that could appear after long periods of time.
   b. Testing the efficiency of the network protocol by sending packets of test to the client and checking for integrity loss.
3. Critical Components Test
   a. Testing the functionality of the component in environments that lack the minimum resource requirements.

LumiereSombre@Team

Contact name Reznicencu Bogdan

Team rep. reznicencu.bogdan99@gmail.com

Repo. Git. https://github.com/ReznicencuBogdan
Telephone no. +40 (744 368 823)