**TRƯỜNG ĐẠI HỌC ĐÀ LẠT**

**KHOA CÔNG NGHỆ THÔNG TIN**

**A picture containing room

Description automatically generated**

**BÁO CÁO MÔN LẬP TRÌNH MẠNG**

**ĐỀ TÀI: XÂY DỰNG ỨNG DỤNG BẮT GÓI TIN**

GVHD: Thầy Trần Ngô Như Khánh

Sinh viên thực hiện: La Quốc Thắng,

Nguyễn Thị Linh,

Nguyễn Thị Bích Ngọc

**Đà Lạt, tháng 06 năm 2020**

# DANH SÁCH THÀNH VIÊN

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Số TT** | **Họ Và Tên** | | **MSSV** | **Lớp** |
| 1 | Nguyễn Thị Bích | Ngọc | 1610171 | CTK40 |
| 2 | La Quốc | Thắng | 1610207 | CTK40 |
| 3 | Nguyễn Thị | Linh | 1610156 | CTK40 |

# BẢNG PHÂN CÔNG CÔNG VIỆC

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Ngày phân công thực hiện** | **Nội dung công việc** | | | **Ghi chú** |
| La Quốc Thắng | Nguyễn Thị Linh | Nguyễn Thị Bích Ngọc |  |
| 1 | 18/04/2020 | Thảo luận chọn đề tài, thống nhất đặt tên ứng dụng là Sniffer | | | Đề tài tự chọn 3: Ứng dụng bắt gói tin |
| 2 | 20/04/2020 | Tìm hiểu thư viện Pcap.Net,  <https://github.com/PcapDotNet/Pcap.Net> | | | Nghiên cứu một số thư viện có sẵn cho C# như Pcap.Net và SharpPcap |
| 3 | 23/04/2020 | Tạo file báo cáo theo mẫu | | |  |
| 4 | 27/04/2020 | Xây dựng GUI cho ứng dụng |  |  | Nêu rõ cho mọi người hiểu được cách code chức năng cho chương trình theo hướng sự kiện |
| 5 | 11/05/2020 | Code chức năng chương trình | | | Ghi một số nội dung vào file báo cáo |
| 6 | 24/05/2020 |  | Code chức năng chương trình | |  |
| 7 | 31/05/2020 |  | Code chức năng chương trình | |  |
| 8 | 07/06/2020 | Cải thiện và sửa một số lỗi |  |  | Hoàn thiện chức năng của chương trình |
| 9 | 12/06/2020 | Hoàn thiện báo cáo đề tài (giữa kỳ) | | |  |
| 10 | 18/06/2020 | Họp nhóm và bắt đầu công việc bổ sung tính năng | | |  |
| 11 | 21/06/2020 | Cập nhật báo cáo đề tài (cuối kỳ) | | |  |

# MÔ HÌNH HOẠT ĐỘNG CỦA CHƯƠNG TRÌNH

Chương trình này áp dụng mô hình 3 tầng nhằm chia tách công việc thành các project khác nhau, đang rất thông dụng hiện nay.

Hình 1. Mô hình hoạt động

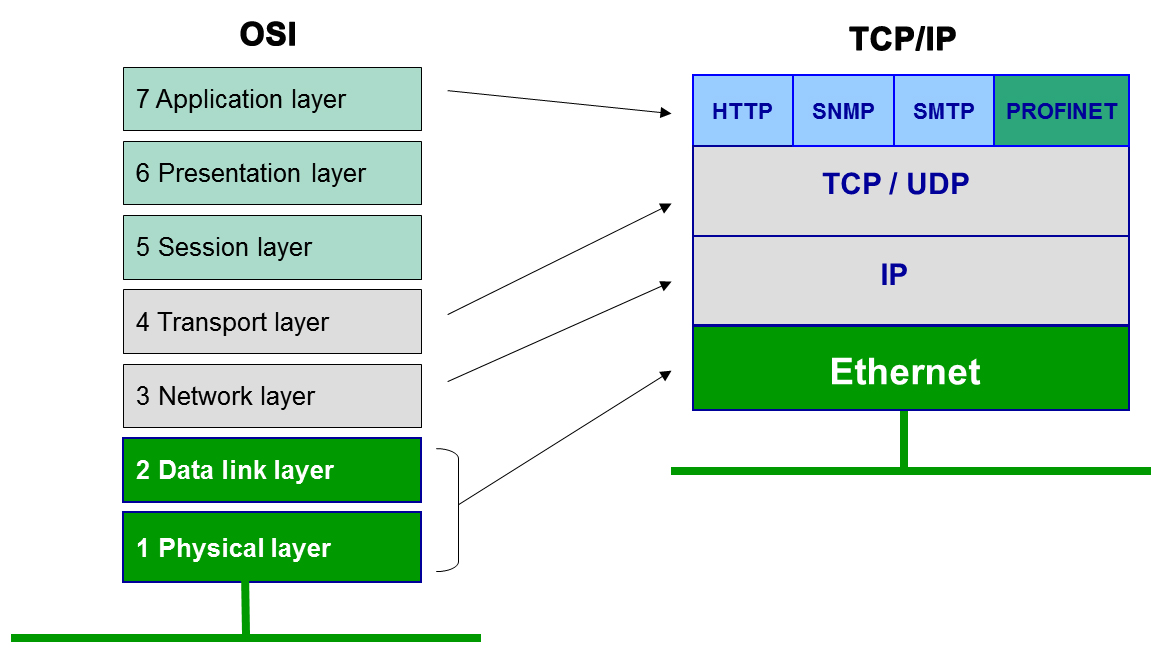
Mô tả mô hình:

* Tầng đầu tiên là *Data Access*, tầng này là tầng thấp nhất của mô hình, nơi mà các tầng cao hơn sử dụng nó để truy cập dữ liệu (khai thác các gói tin trong mạng). Trong phạm vi đề tài này, tầng Data Access bao gồm hai thành phần chính là WinPcap và Pcap.Net. Pcap.Net là thư viện mã nguồn mở, được xây dựng thành Nuget Package và quản lý bởi nuget.org, còn WinPcap cung cấp trình điều khiển (driver) để Pcap.net hoạt động được.
* Tầng *Business* còn gọi là tầng nghiệp vụ, thuộc loại Class Library Project, sử dụng khả năng liệt kê các interface mạng và bắt gói tin của thư viện Pcap.Net để xây dựng các phương thức cho tầng Application sử dụng.
* Tầng *Application* được gọi là tầng ứng dụng, thông thường nó có thể là Windows Forms (WinForms), Windows Presentation Foundation (WPF), Mobile App, Website,… sử dụng các phương thức được định nghĩa trong tầng Business mà không cần truy cập trực tiếp tầng Data Access, cung cấp giao diện độ họa cho người dùng sử dụng.

Một số lợi ích của mô hình 3 tầng này:

* Phát triển các thành phần/module một cách độc lập.
* Dễ bảo trì phần mềm, không ảnh hưởng đến các lớp khác thuộc tầng khác.
* Tầng cao hơn chỉ cần sử dụng các phương thức của tầng thấp hơn mà không cần quan tâm đến thực hiện những gì.
* Dễ phát triển, nâng cấp và tái sử dụng cho sau này.
* Dễ phân chia công việc cho các thành viên.

Bên cạnh đó, từ mô hình TCP/IP, nhóm sẽ trích xuất một số thông tin của mỗi tầng bên trong gói tin và hiển thị lên khi người dùng chọn gói tin đó.



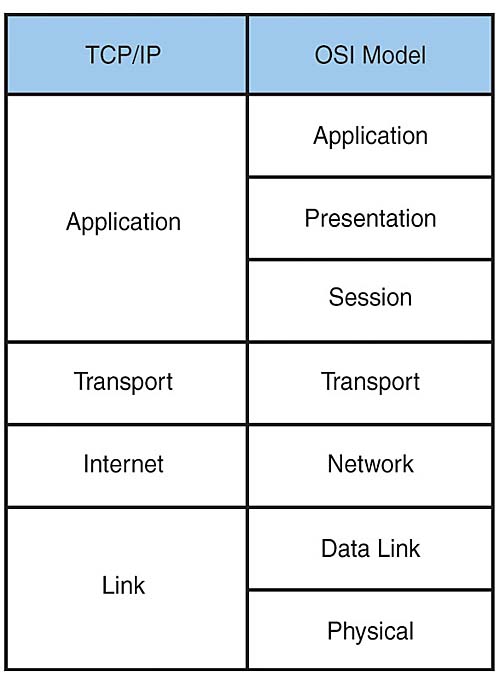
Hình . Các tầng theo mô hình TCP/IP

# THƯ VIỆN SỬ DỤNG

## WinPcap

### Giới thiệu

Nhiều năm nay, WinPcap được biết đến như là một công cụ chuẩn công nghiệp cho việc truy cập *tầng liên kết* của mạng trong môi trường Windows, cho phép các ứng dụng có thể bắt và truyền gói tin mà bỏ qua ngăn xếp giao thức.

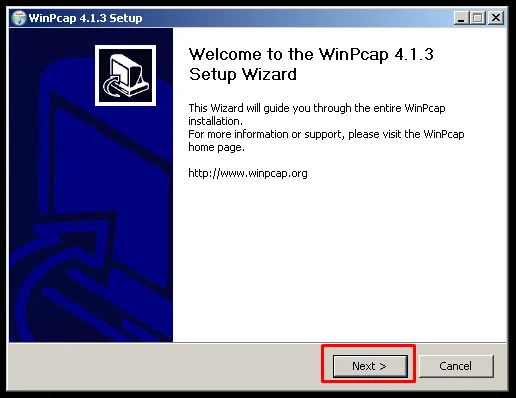


Hình . Tầng Liên kết trong mô hình TCP/IP

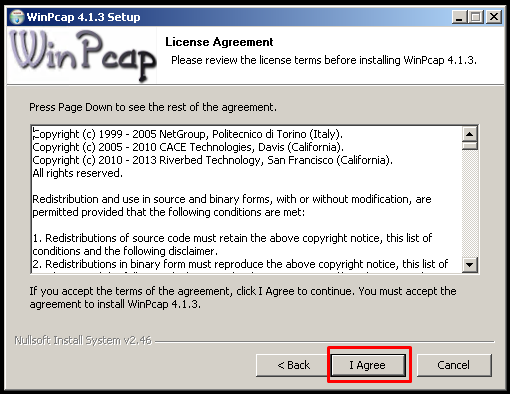
WinPcap bao gồm một trình điều khiển mở rộng khả năng của hệ điều hành trong truy cập mạng ở tầng thấp (tầng liên kết) và một thư viện để có thể sử dụng dễ dàng hơn.

### Cài đặt

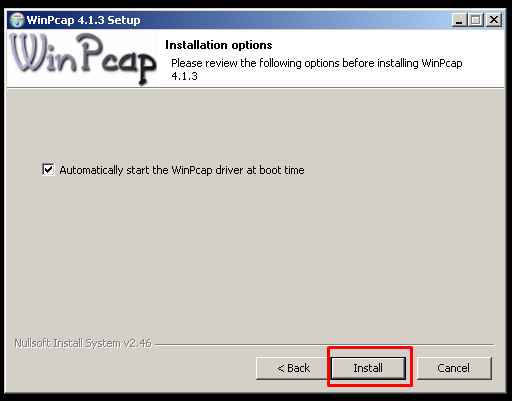
Để sử dụng được Pcap.Net thì trước hết phải tải về WinPcap tại địa chỉ sau <https://www.winpcap.org/install/default.htm> và cài đặt nó.



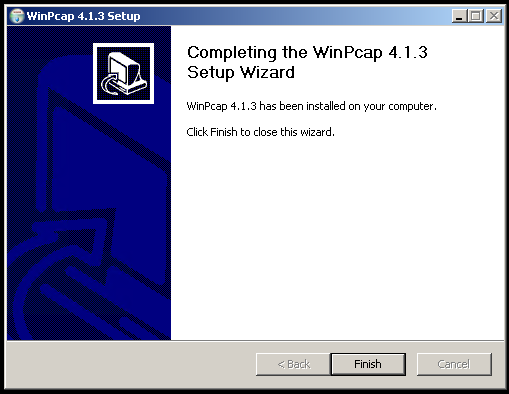
Hình . Giao diện cài đặt WinPcap



Hình . Đồng ý các điều khoản của chương trình



Hình . Chấp nhận WinPcap tự động khởi động lúc bật máy



Hình . WinPCap cài đặt thành công

## Pcap.Net

### Giới thiệu

Pcap.Net thực chất là một lớp bao bọc WinPcap vốn được viết bằng ngôn ngữ C/C++, sau được viết ra nhằm hỗ trợ trong lập trình .NET. Pcap.Net có hầu hết các tính năng của WinPcap và bao gồm một framework phân tích các gói tin.

Một số tính năng đáng chú ý của Pcap.Net:

* Lấy danh sách các thiết bị mạng (interface hay còn gọi là adapter) của máy tính.
* Đọc các gói tin trực tiếp từ thiết bị mạng hoặc từ tập tin.
* Nhận thống kê toàn bộ gói tin được bắt hoặc một số gói tin.
* Hỗ trợ nhiều phương thức khác nhau.
* Áp dụng bộ lọc Berkeley Packet[[1]](#footnote-2).
* Gửi các gói tin trực tiếp đến thiết bị mạng.
* Lưu các gói tin vào file.
* Sử dụng Enumerables (và LINQ) để nhận các gói tin.

Ngoài ra, Pcap.Net còn có thể phân tích các gói tin sau:

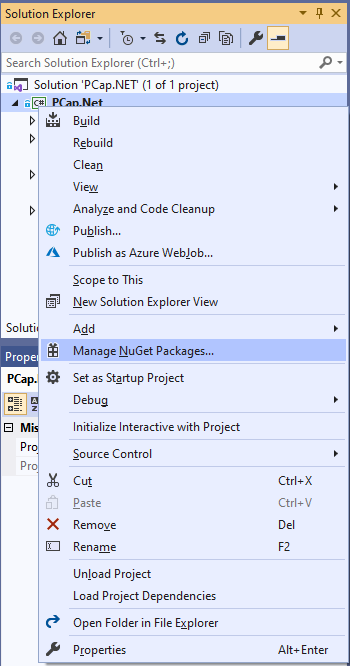
* Ethernet + VLAN tagging (802.1Q)
* ARP
* IPv4
* IPv6
* GRE
* ICMP
* IGMP
* UDP
* TCP
* DNS
* HTTP

Tuy nhiên, do phạm vi đề tài này cũng như các tài liệu hướng dẫn khác nên chỉ tập trung xử lý một vài gói tin thông dụng như TCP, UDP, ARP.

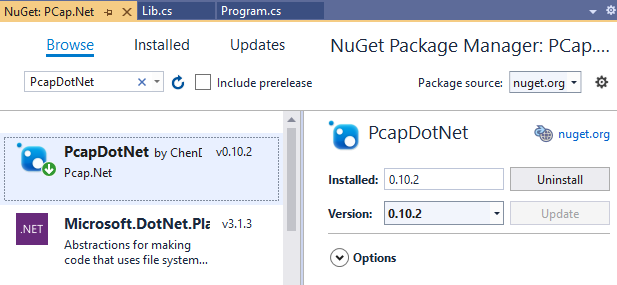
### Sử dụng Pcap.Net trong Visual Studio

Để dự án có thể sử dụng được Pcap.Net thì trong Microsoft Visual Studio, ta thực hiện các bước sau:

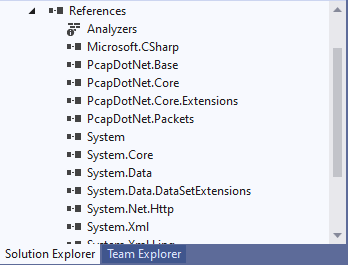
* Cài đặt phiên bản .NET Framework 4.5 trở lên
* Tạo một dự án Console, Windows Form hay Class Library
* Thêm thư viện PcapDotNet vào project với các bước như sau:



Hình . Nhấp chuột phải vào project, chọn Manage Nuget Packages…



Hình . Vào Browse, gõ PcapDotNet và nhấn Install (nằm bên phải)



Hình . Các thư viện tham chiếu của PcapDotNet sẽ xuất hiện trong project

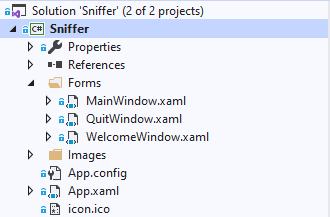
Pcap.Net còn cung cấp một số hướng dẫn sử dụng Pcap.Net để bắt gói tin, phân tích gói tin, lưu gói tin, mở gói tin và lọc các gói tin. Chúng ta có thể truy cập hướng dẫn này tại đường dẫn sau <https://github.com/PcapDotNet/Pcap.Net/wiki/Pcap.Net-Tutorial>.

# MÔ TẢ CHƯƠNG TRÌNH

## Tầng Application

### Giao diện

Tầng Application chứa dự án WPF có nhiệm vụ phát triển ứng dụng có giao diện đồ họa (GUI) cho phép người dùng thao tác trực tiếp trên đó.

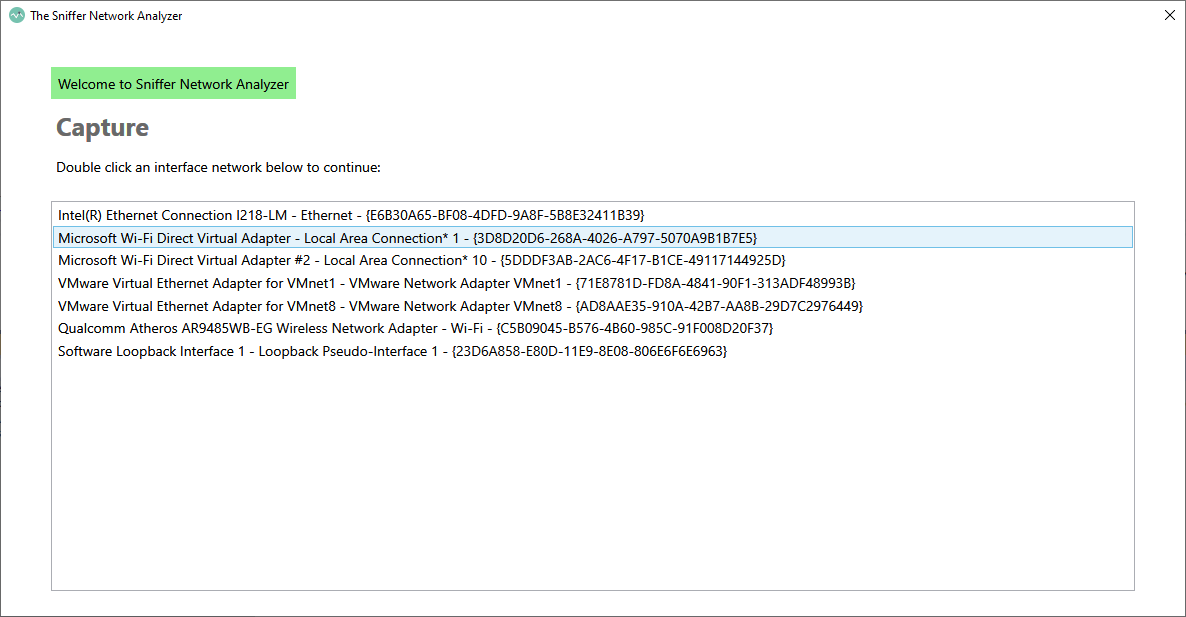


Hình . Bên trong tầng Application

Thư mục Forms tập hợp các form (các cửa sổ) có trong ứng dụng. Phần mở rộng của form trong WPF là .xaml. XAML là chữ viết tắt của *eXtensible Application Markup Language*, là thế hệ form tiếp theo của Microsoft nhằm thay thế Windows Forms truyền thống, dựa trên XML để mô tả giao diện người dùng đồ họa (GUI).

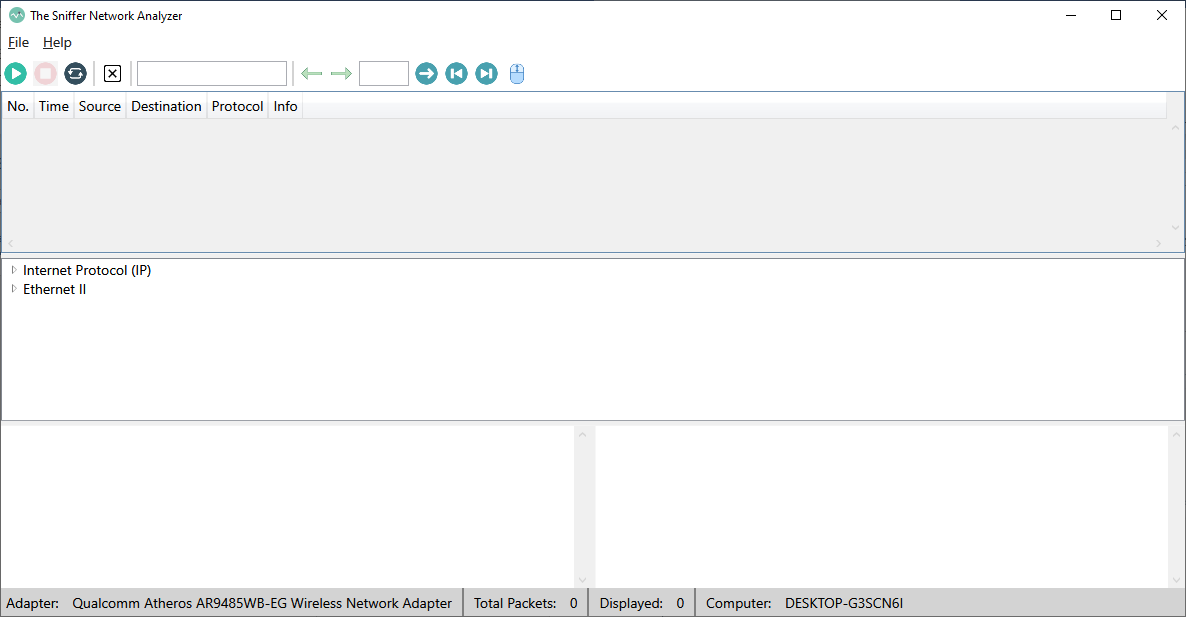
Các form có trong chương trình:

* *WelcomeWindow*: Form xuất hiện lần đầu khi mở chương trình, cho phép người dùng chọn một card mạng (interface) từ danh sách. Tất cả các interface có trong máy sẽ được hiển thị ở đây.



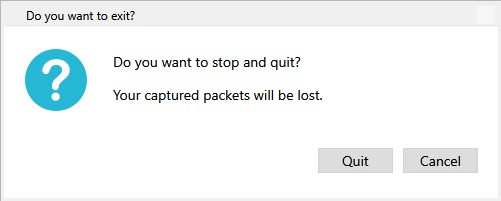
Hình . Form WelcomeWindow

* *MainWindow*: Sau khi người dùng chọn một interface mạng ở WelcomeWindow, một form mới mở ra chứa các chức năng chính để người dùng có thể thao tác với chương trình.



Hình . Form MainWindow

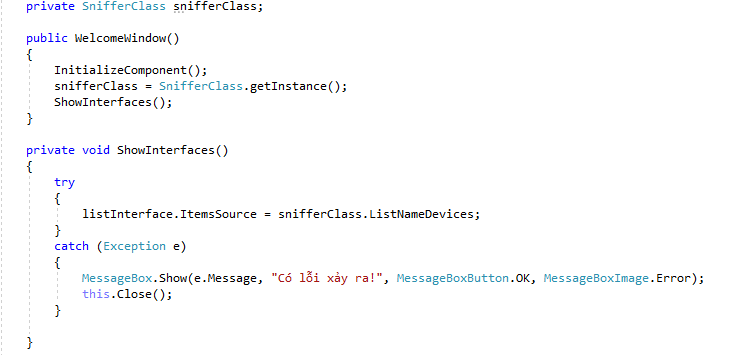
* *QuitWindow*: Sau khi thao tác xong ở form MainWindow, nếu người dùng nhấn nút **X** để thoát, một form mới xuất hiện, yêu cầu xác thực của người dùng có muốn thoát hay không.



Hình . Form QuitWindow

### Xử lý sự kiện

1. Form WelcomeWindow

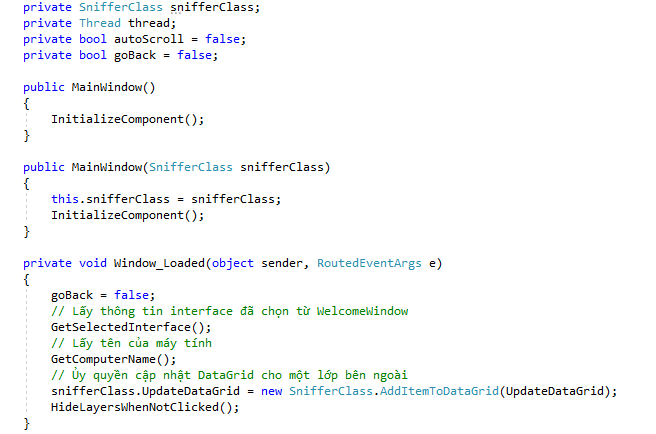


Hình . Lấy và hiển thị danh sách interface ngay sau khi form xuất hiện

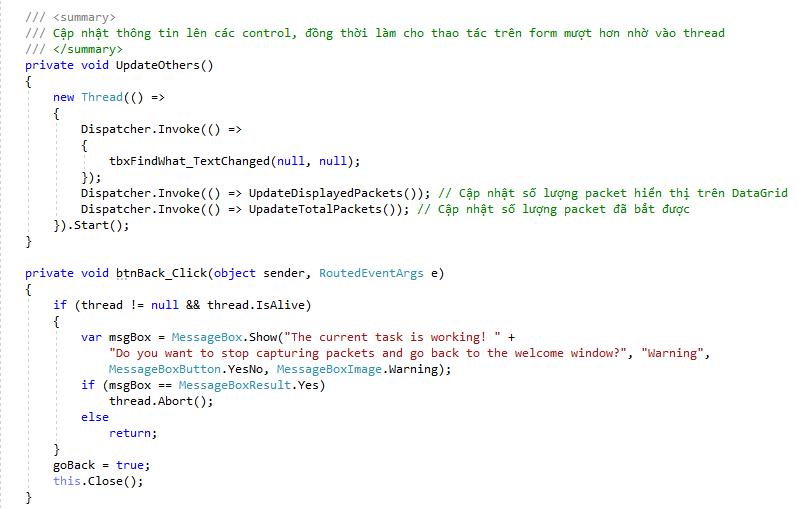


Hình . Sự kiện nhấn đúp chuột hoặc nhấn phím Enter sẽ chọn interface

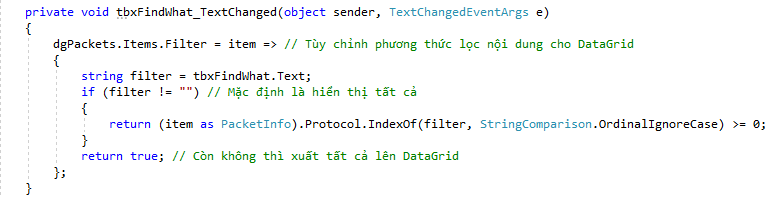
1. Form MainWindow



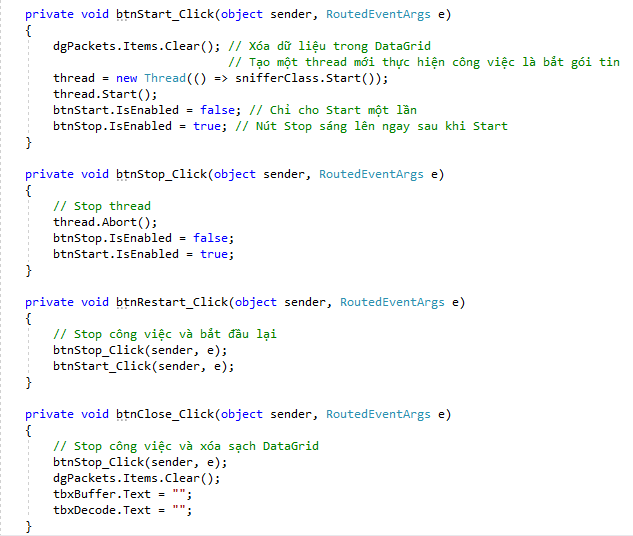
Hình . Thực hiện một số công việc ngay sau khi form hiện trên màn hình



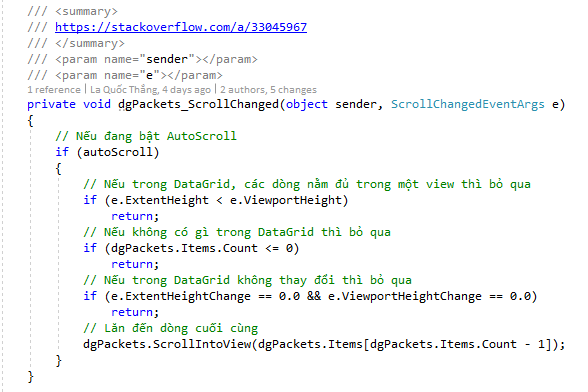
Hình . Các thao tác xảy ra khi một gói tin bắt được



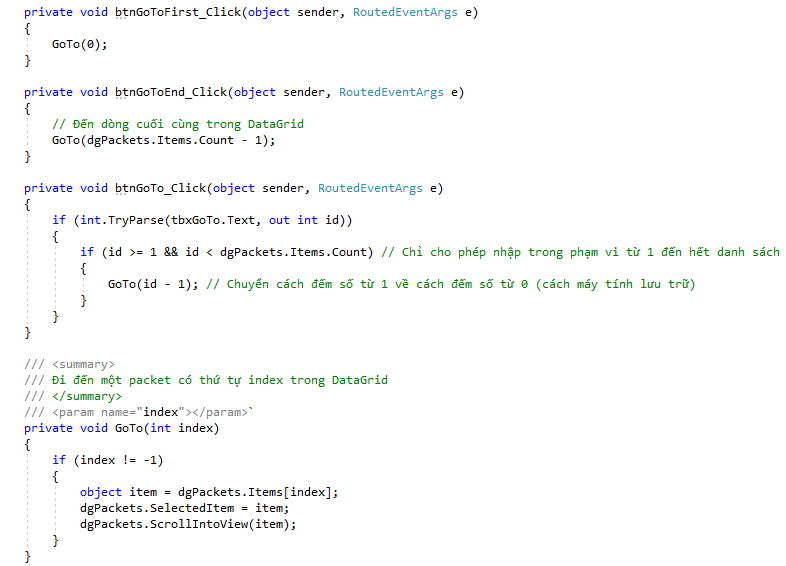
Hình . Lọc các gói tin



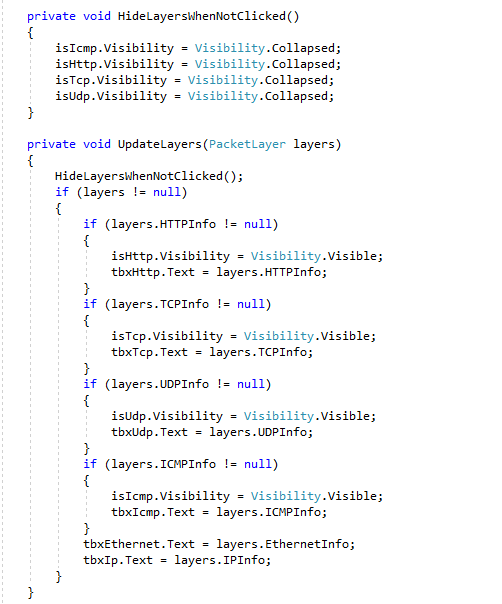
Hình . Chức năng Start, Stop và Restart bắt gói tin



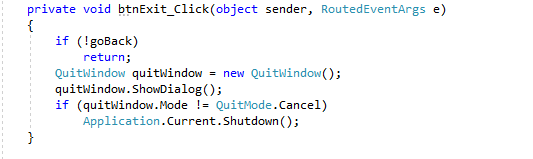
Hình . Chức năng tự động lăn xuống dòng cuối cùng trong DataGrid



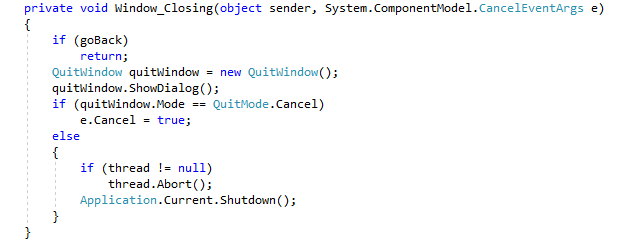
Hình . Đi đến gói tin nằm đầu, cuối và bất kỳ trong DataGrid



Hình . Xuất thông tin mỗi tầng (nếu có)

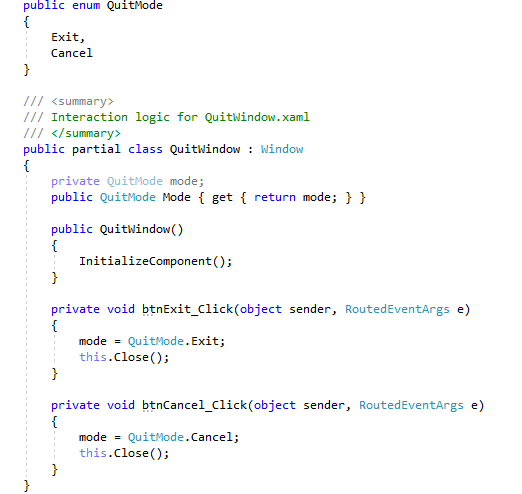


Hình . Cho phép quay về màn hình chính hay thoát hoàn toàn ứng dụng



Hình . Tương tự như vậy, thêm xử lý vào sự kiện thoát khỏi chương trình

1. Form QuitWindow

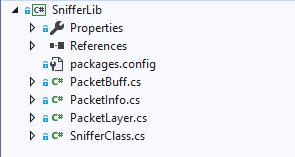


Hình . Sự kiện bên trong QuitWindow

## Tầng Business

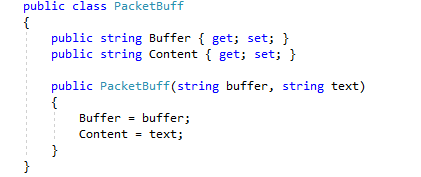
Tầng Business trong đề tài có tên là SnifferLib, nó thuộc dạng Class Library (thư viện lớp). Bên cạnh các project có dạng Windows Application, Console Application thì Class Library tập hợp các mẫu (template) trong lập trình hướng đối tượng. Class Library chứa các mã được định nghĩa trước, các project khác có thể tái sử dụng các mã này trong project của mình, sử dụng như một tham chiếu (reference). Bên trong Class Library có thể chứa mã dành cho xử lý các thành phần giao diện (nút, biểu tượng, cửa sổ, datagrid,…) và phi giao diện (các chức năng khác). Mọi ngôn ngữ lập trình hướng đối tượng đều có sẵn các thư viện như thế này, được đi kèm trong ngôn ngữ đó và người lập trình có thể định nghĩa riêng cho mình.

Khi build chương trình thì các dự án dạng Class Library này sẽ trở thành các tập tin .dll ( Dynamic Link Library).



Hình . Tầng Business của ứng dụng này chính là SnifferLib

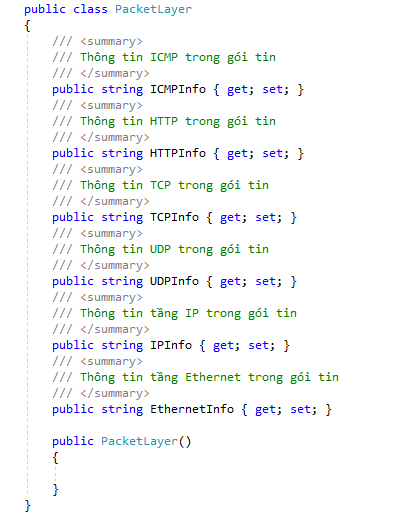
### PacketBuff



Hình . Lớp PacketBuff

Lớp PacketBuff có hai thuộc tính là Buffer chứa nội dung của gói tin ở dạng thập lục phân (hexadecimal), còn Content là thông tin đã được chuyển thành chuỗi Unicode từ Buffer.

### PacketLayer



Hình . Lớp PacketLayer

Lớp PacketBuffer bao gồm các thuộc tính là nội dung của mỗi tầng trong mô hình TCP/IP của một gói tin.

### PacketInfo

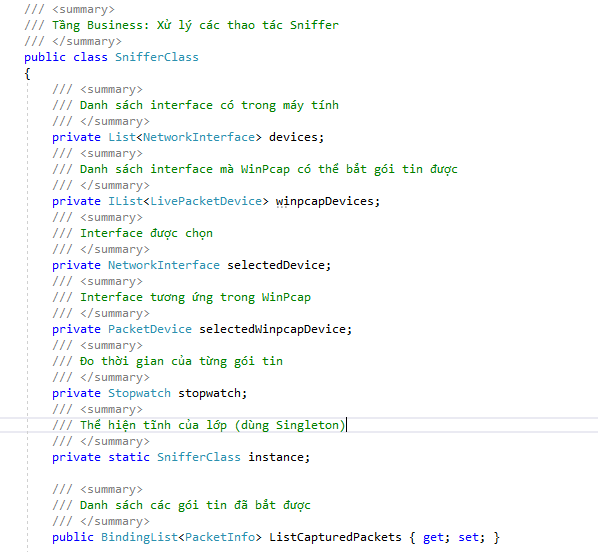


Hình . Lớp PacketInfo chứa một số thuộc tính cần thiết để hiển thị trên DataGrid

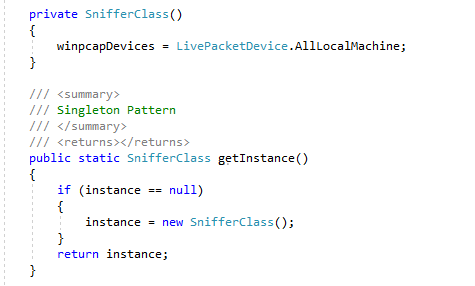
Bên trong lớp PacketInfo có định nghĩa một số thuộc tính của gói tin, trong đó có sử dụng PacketBuffer và PacketLayer. Vì bên trong gói tin có chứa nhiều thông tin, nên trước khi đưa vào đây thì các thông tin đó phải được xử lý. Công việc xử lý là của lớp SnifferClass.

### SnifferClass

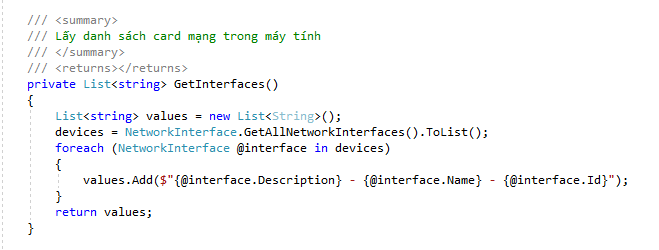
SnifferClass là lớp trọng yếu của tầng Business vì nó trực tiếp xử lý các gói tin bắt được thông qua một số hàm từ thư viện Pcap.Net. Như vậy, SnifferClass có chức năng liên kết giữa tầng Pcap.Net với tầng Application, giúp đảm bảo cho các form ở tầng Ứng dụng hoạt động được.



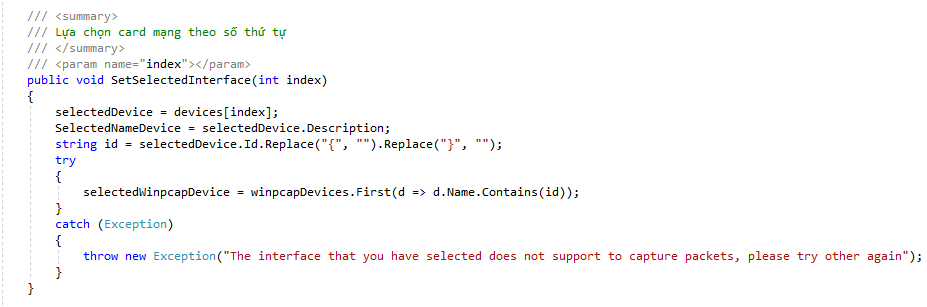
Hình . Một số trường (fields) và thuộc tính (properties) của SnifferClass



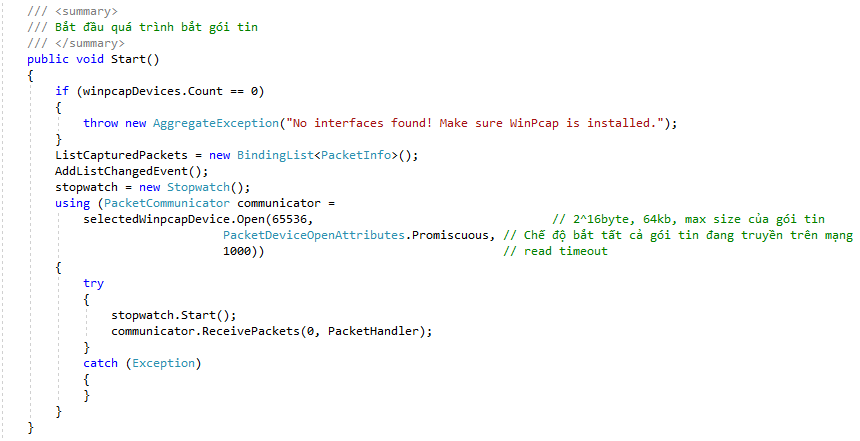
Hình . Sử dụng Singleton để lớp SnifferClass chỉ khởi tạo một lần



Hình . Phương thức lấy các interface có trong máy tính



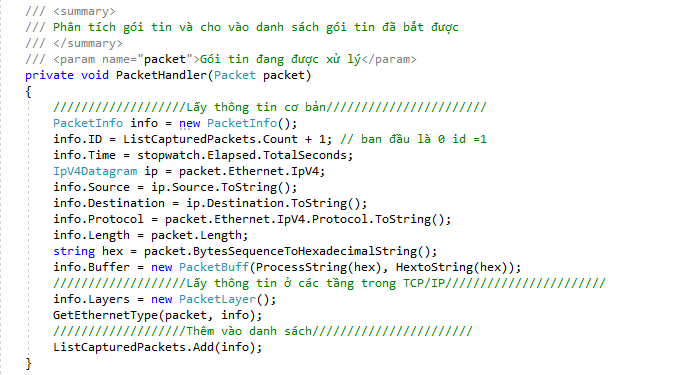
Hình . Đối chiếu interface được chọn với interface trong Pcap.Net



Hình . Phương thức khởi động quá trình bắt gói tin

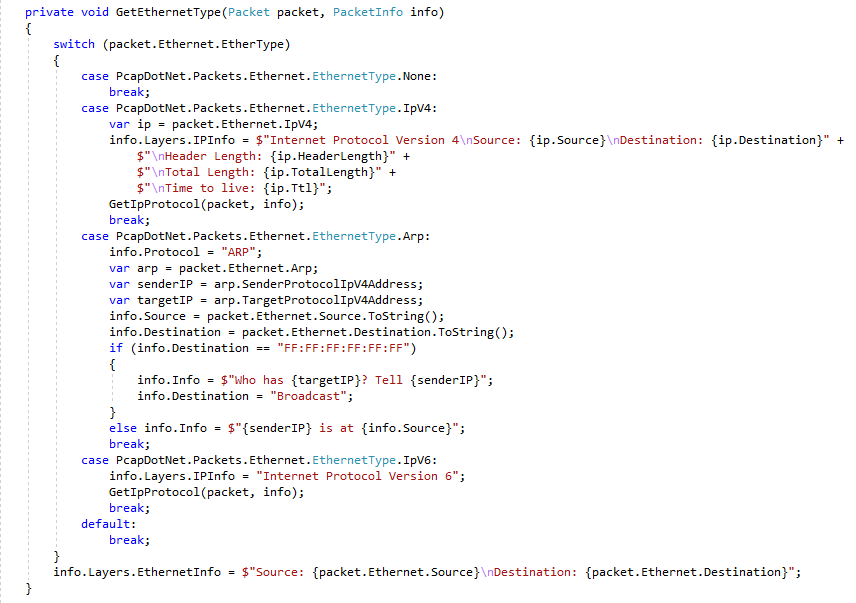
Open() mở một adapter ghi nhận lại lưu lượng đang truyền trong mạng với các tham số:

* **snapshotLength**: Chỉ định phần của gói sẽ được bắt lấy. Ở một số hệ điều hành, trình điều khiển gói tin có thể được cấu hình để bắt một phần gói tin, làm giảm bộ nhớ dữ liệu của ứng dụng, cải thiện hiệu suất của phương thức. Tuy nhiên, giá trị 65536 được sử dụng trong đây là kích thước tối đa của gói tin có thể có, tương đương 64KB. Điều này đảm bảo sẽ nhận được toàn bộ gói tin.
* **attributes**: Ở chế độ bình thường, adapter chỉ nhận các gói tin từ mạng được gửi đến cho nó, các gói tin trao đổi khác sẽ bị bỏ qua. Còn trong chế độ ngẫu nhiên (promiscuous), adapter sẽ bắt tất cả các gói tin cho dù nó có dự định gửi đến chính thiết bị đó hay không.
* **readTimeOut**: Thời gian chờ để đọc, tính bằng mili-giây. Mỗi lần adapter đọc, nó sẽ luôn trả về cái gì đó sau mỗi readTimeOut, dù có hay không gói tin. Nếu đặt giá trị 0, adapter sẽ không trả về nếu không có gói tin đến, -1 nghĩa là luôn trả về ngay lập tức.

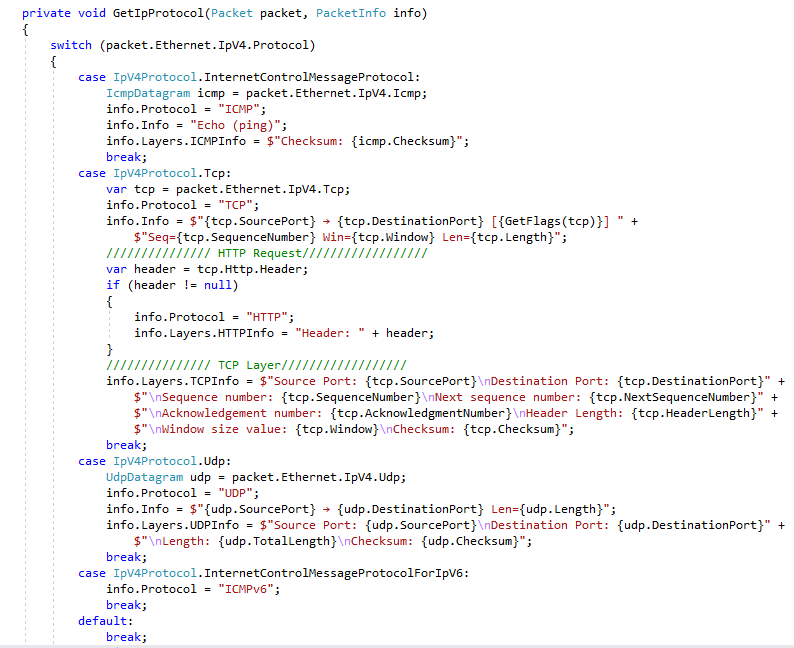


Hình . Phân tích gói tin đã bắt được, chuyển nó thành PacketInfo, thêm vào danh sách

Để chi tiết hơn thông tin của gói tin, ngoài các thông tin như IP nguồn, IP đích, độ dài gói tin, giao thức sử dụng thì thông tin mô tả cho nó cũng khá là quan trọng. Nếu gói tin là ARP thì cần phải thể hiện thông tin là nó đang đi tìm địa chỉ MAC của một máy có IP này, chẳng hạn vậy.

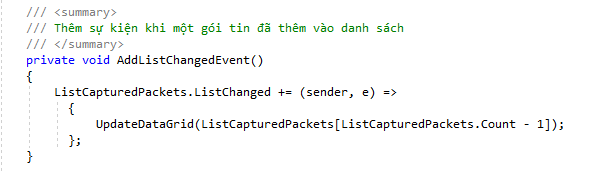


Hình . Lấy thông tin cần thiết ở tầng Ethernet



Hình . Lấy thông tin ở tầng IP, tầng TCP/UDP và các giao thức ICMP, HTTP

Để có thể chủ động mỗi khi bắt được một gói tin và cập nhật vào DataGrid thì cần phải định nghĩa sự kiện danh sách thay đổi (list changed event). Khi một PacketInfo thêm vào danh sách, nó sẽ gọi UpdateDataGrid (hàm mà bên form đã ủy quyền cho nó) và thêm cái PacketInfo cuối cùng của danh sách vào DataGrid.



Hình . Sự kiện cập nhật PacketInfo tron DataGrid thêm một gói tin bắt được

## Tầng Data Access

Để nói về phần này, trước hết chúng ta sẽ tìm hiểu về trình điều khiển NPF (Netgroup Packet Filter), đây là kernel quan trọng của WinPcap. Đối với người sử dụng thông thường, chúng ta sẽ chỉ quan tâm đến cách sử dụng WinPcap như thế nào, còn các nhà phát triển, các lập trình viên có thể quan tâm đến cấu trúc của nó, cách hoạt động.

NPF là thành phần của WinPcap làm những công việc “quan trọng nhất”, đó là xử lý các gói tin truyền trong mạng và xuất các gói tin đã bắt được, đưa đến cho người dùng.

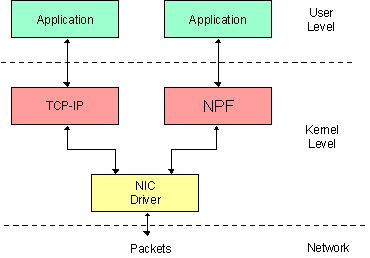
Pcap.Net sử dụng NPF của WinPcap để lấy về các gói tin như vậy.

### NPF và NDIS

NDIS (Đặc tả giao diện trình điều khiển mạng - Network Driver Interface Specification) là một tiêu chuẩn xác định giao tiếp giữa adapter (hoặc trình điều khiển của adapter) và trình điều khiển giao thức (ví dụ như TCP/IP). Mục đích chính của NDIS là hoạt động như một trình bao bọc cho phép các trình điều khiển giao thức gửi và nhận các gói tin trong mạng (LAN hoặc WAN) mà không cần quan tâm đến bộ điều hợp cụ thể hoặc hệ điều hành Win32 cụ thể.

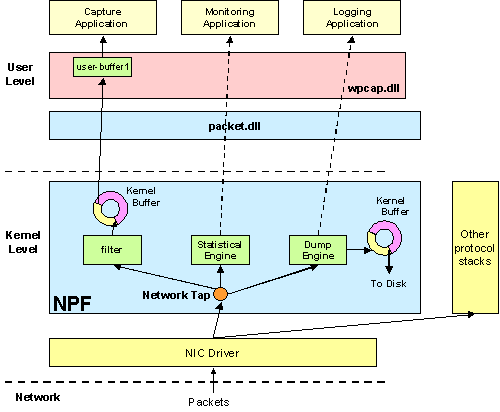
NDIS hỗ trợ 3 loại trình điều khiển mạng:

* **Network interface card (NIC)**: Trình điều khiển NIC trực tiếp quản lý các NIC, cung cấp interface giúp các tầng ở trên phần cứng có thể gửi gói tin, xử lý ngắt, cài đặt lại NIC, tạm dừng,… Trình điều khiển NIC có thể là miniport hoặc legacy.
* **Các trình điều khiển trung gian**: Trình điều khiển trung gian nằm giữa trình điều khiển giao thức và miniport. Nó còn có thể là tầng trên của trình điều khiển trung gian khác mặc dù việc xếp lớp như vậy có thể có ảnh hưởng tiêu cực đến hiệu năng hệ thống. Nó có nhiệm vụ thực hiện dịch phương tiện từ giao thức này sang giao thức kia.
* **Trình điều khiển vận chuyển hoặc giao thức**: Thực hiện một ngăn xếp giao thức mạng như IPX/SPX hoặc TCP/IP, cung cấp dịch vụ cho một hoặc nhiều NIC. Nó phục vụ cho tầng ứng dụng (tầng cao hơn nó) và kết nối đến một hoặc nhiều NIC, NDIS trung gian khác ở tầng thấp hơn.



Hình . NPF nằm trong NDIS

### Cấu trúc cơ bản của NPF



Hình . Trình điều khiển thiết bị NPF

NPF có thể thực hiện một số công việc như bắt gói tin, theo dõi, lưu vào đĩa, tiêm gói tin (packet injection).

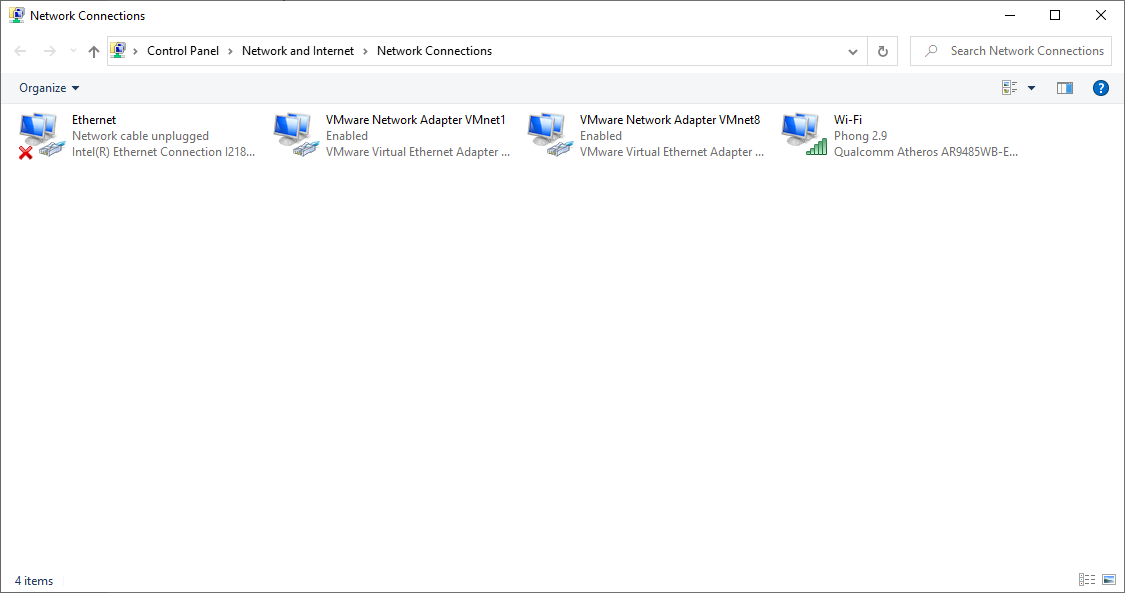
Về bắt gói tin, đây là công việc quan trọng nhất của NPF. Trong suốt quá trình bắt gói tin, trình điều khiển “đánh hơi” (sniff) các gói tin bằng cách sử dụng một interface mạng và đưa nó cho ứng dụng bên ngoài sử dụng.

Quá trình bắt gói tin phụ thuộc vào hai thành phần chính:

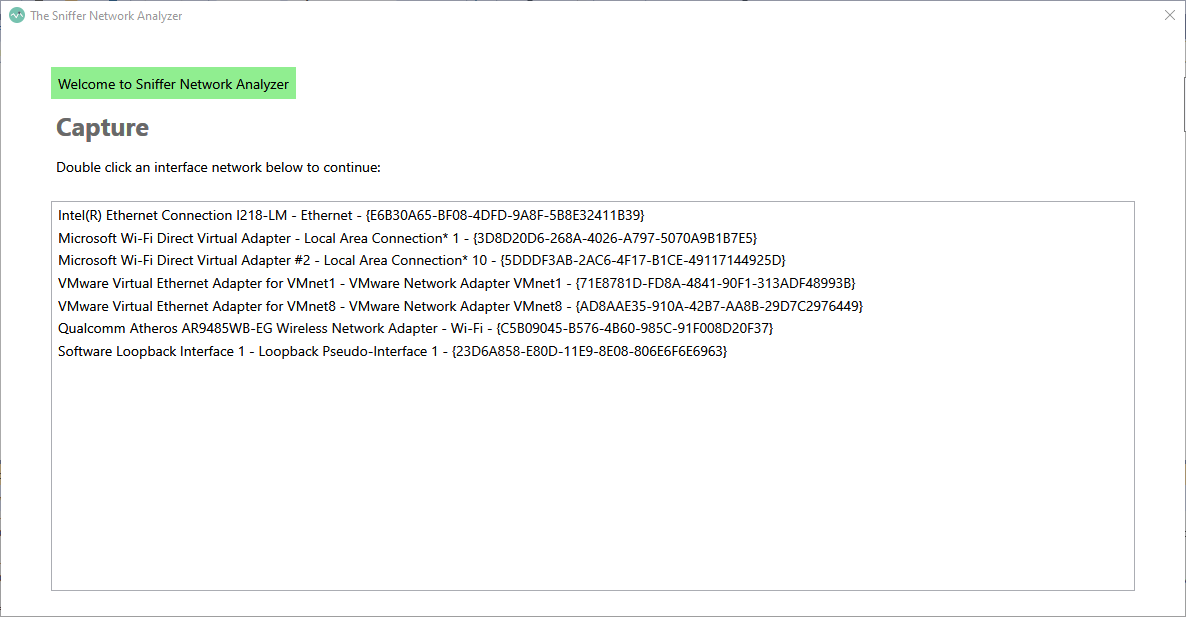
* Một bộ lọc quyết định một gói tin đang đến có được chấp nhận để sao chép nó đến cho ứng dụng đang chờ. Hầu hết các ứng dụng sử dụng NPF từ chối hơn là chấp nhận, do đó bộ lọc gói linh hoạt và hiệu quả mang lại hiệu suất cao cho ứng dụng. Bộ lọc này được dẫn xuất từ Berkeley Packet Filter[[2]](#footnote-3) (BPF).
* Một bộ đệm lưu trữ gói tin để tránh thất thoát. Nó lưu lại cùng với header bao gồm dấu thời gian (timestamp), kích thước gói tin. Nếu độ đệm bị đầy thì khi một gói tin đến, gói tin đó sẽ bị loại bỏ. Tuy nhiên, cả bộ đệm người dùng và bộ đệm kernel có thể được thay đổi trong thời gian chạy để có thể linh hoạt tối đa.

Các chức năng khác như tiêm gói tin, theo dõi mạng, lưu vào đĩa sẽ không đề cập tại đây.

# KẾT QUẢ THỰC HIỆN (DEMO)



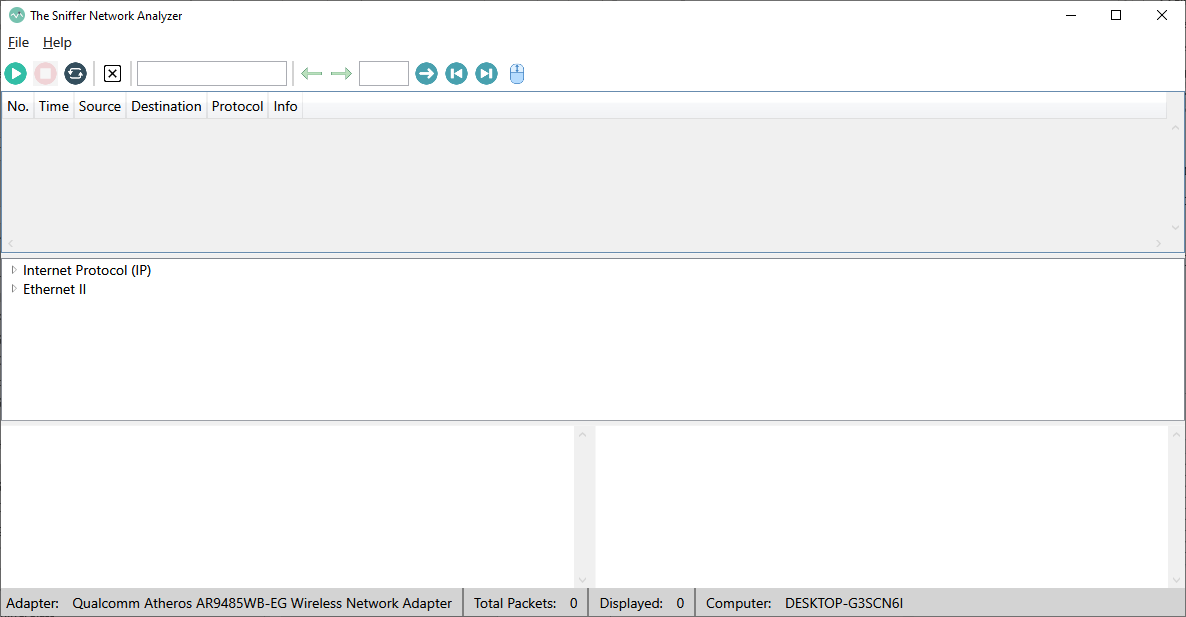
Hình . Danh sách các interface mạng (adapter) trong máy tính



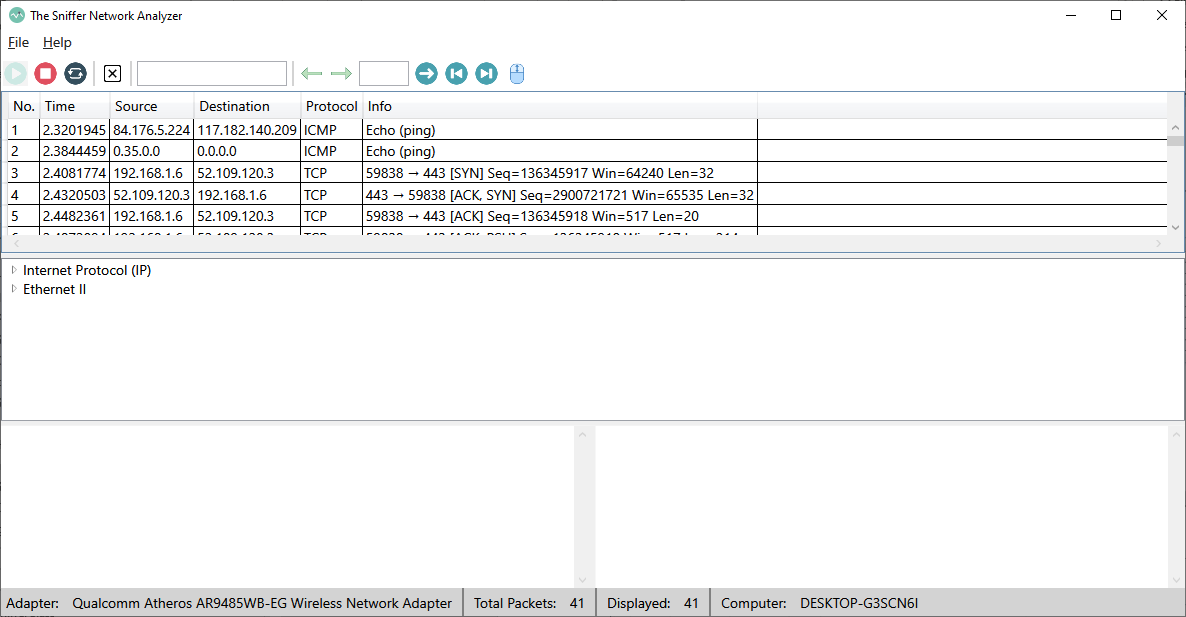
Hình . Danh sách các interface mạng hiển thị trên form

Sau khi chọn xong, chương trình sẽ mở form chính để ta có thể thao tác. Có thể chia cửa sổ thành 5 khu vực như sau:

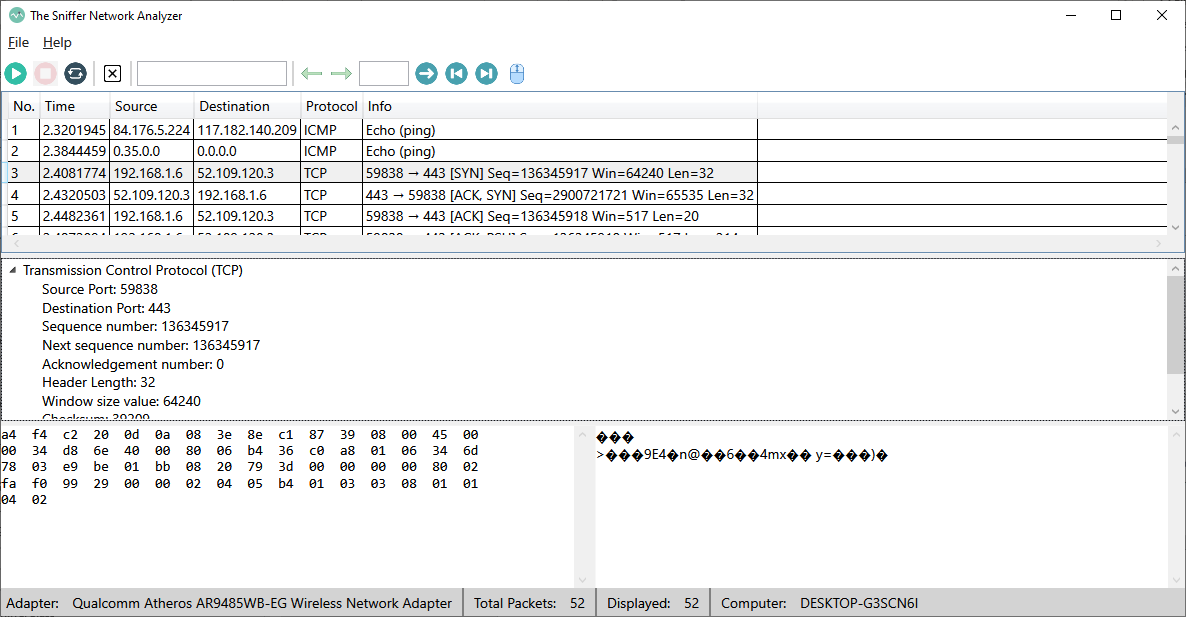
* Thanh công cụ: Start, Stop, Restart, Close, các tính năng khác như tìm kiếm, di chuyển lựa chọn giúp ta truy cập và tìm kiếm các gói tin dễ dàng hơn.
* Danh sách các gói tin với các thông tin: Số thứ tự, Thời gian, IP nguồn, IP đích, Giao thức, Thông tin.
* Thông tin các tầng: Mặc định thì sẽ có hai tầng Ethernet II và Internet Protocol sẽ hiển thị, các tầng các thì tùy thuộc vào từng gói tin tương ứng.
* Nội dung của gói tin: Nội dung được chia thành hai phần với hai định dạng là Thập lục phân và Chuỗi Unicode. Tuy nhiên, nội dung chuỗi nằm trong phần Content không thể hiển thị đúng ký tự.
* Thanh trạng thái: Tên adapter, tổng số gói tin đã bắt được, tổng số gói tin hiển thị trong danh sách, tên của máy tính.



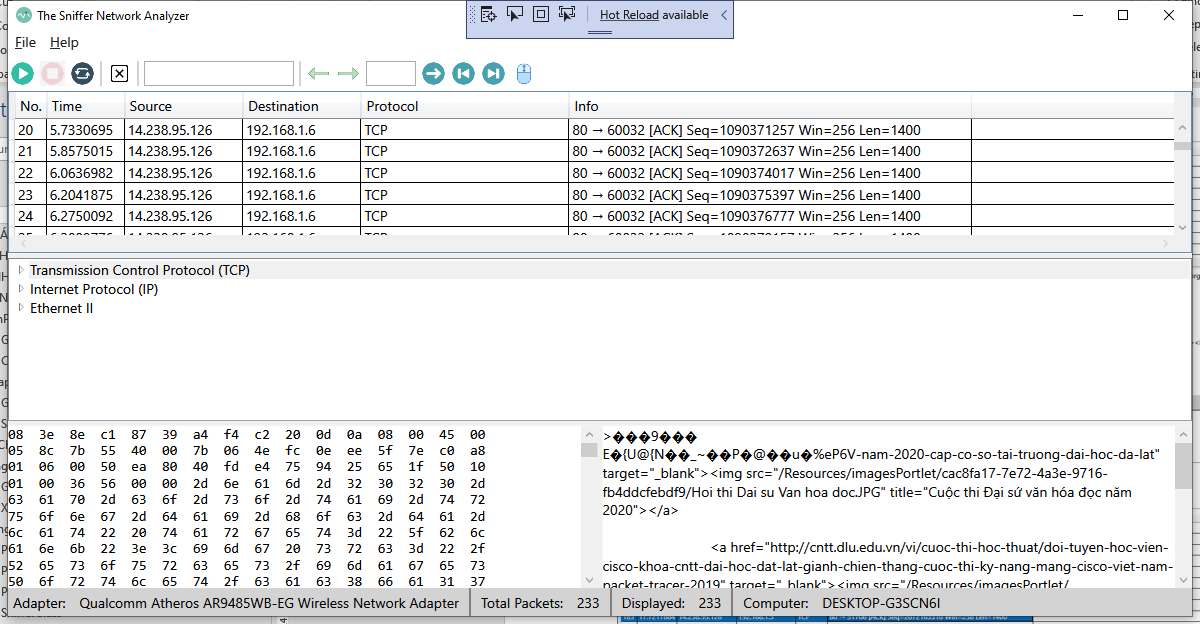
Hình . Giao diện chính của ứng dụng



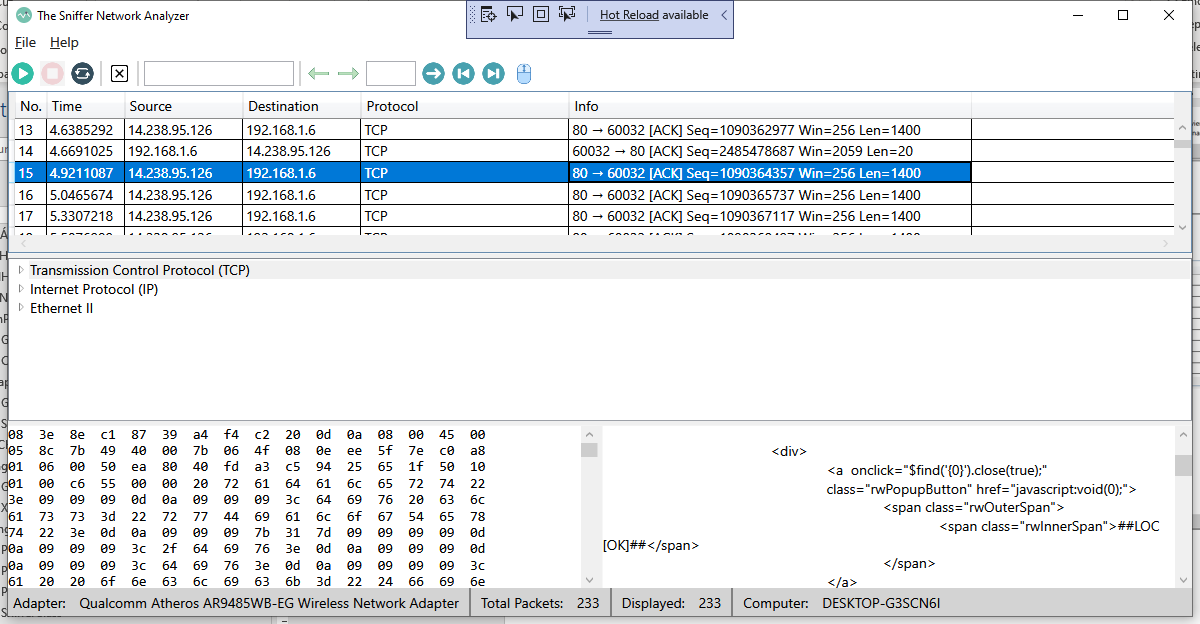
Hình . Kết quả sau khi nhấn nút Start



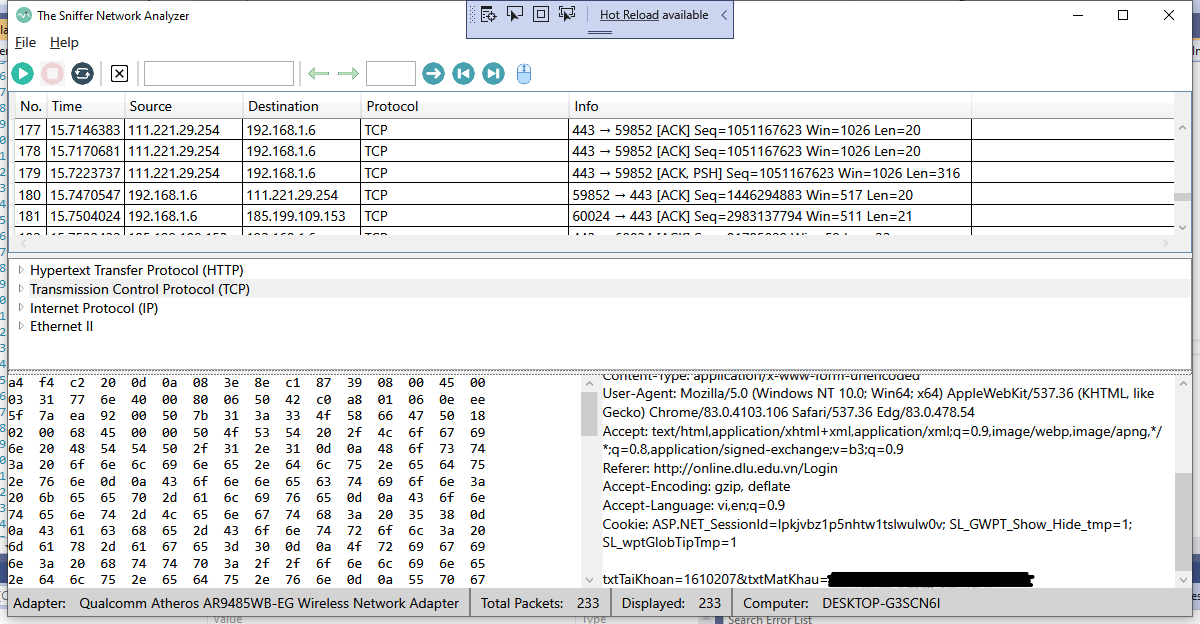
Hình . Nội dung của gói tin sẽ hiện lên



Hình 47. Ngoài ra có thể xem được nội dung website cntt.dlu.edu.vn



Hình 48. Và cả đoạn mã Javascript nữa



Hình 49. Thông tin đăng nhập chưa mã hóa tại trang online.dlu.edu.vn/login

# TÀI LIỆU THAM KHẢO

[1] *Pcap.Net Tutorial*. <https://github.com/PcapDotNet/Pcap.Net/wiki/Pcap.Net-Tutorial>

[2] *Trang chủ WinPcap*. <https://www.winpcap.org/>

[2] *PcapDotNet/Pcap.Net Repo*. <https://github.com/PcapDotNet/Pcap.Net>

[3] *Mô hình 3 lớp (3-layer) có gì hay?* <https://techtalk.vn/mo-hinh-3-lop-co-gi-hay.html>

[4] *XAML là gì?* <https://www.wpf-tutorial.com/vi/5/xaml/xaml-la-gi/>

[5] *XAML overview in WPF.* <https://docs.microsoft.com/en-us/dotnet/desktop-wpf/fundamentals/xaml>

[6] C*lass Library*. <https://www.techopedia.com/definition/3764/class-library#:~:text=A%20class%20library%20is%20a%20pre%2Dcoded%20object%2Doriented%20programming,as%20other%20non%2DGUI%20components.>

[7] *NPF driver internals manual (WinPcap internals)*. <https://www.winpcap.org/docs/docs_412/html/group__NPF.html>

# MỤC LỤC

[DANH SÁCH THÀNH VIÊN 1](#_Toc43667806)

[BẢNG PHÂN CÔNG CÔNG VIỆC 2](#_Toc43667807)

[MÔ HÌNH HOẠT ĐỘNG CỦA CHƯƠNG TRÌNH 4](#_Toc43667808)

[THƯ VIỆN SỬ DỤNG 6](#_Toc43667809)

[I. WinPcap 6](#_Toc43667810)

[1. Giới thiệu 6](#_Toc43667811)

[2. Cài đặt 6](#_Toc43667812)

[II. Pcap.Net 8](#_Toc43667813)

[1. Giới thiệu 9](#_Toc43667814)

[2. Sử dụng Pcap.Net trong Visual Studio 10](#_Toc43667815)

[MÔ TẢ CHƯƠNG TRÌNH 12](#_Toc43667816)

[I. Tầng Application 12](#_Toc43667817)

[1. Giao diện 12](#_Toc43667818)

[2. Xử lý sự kiện 14](#_Toc43667819)

[II. Tầng Business 20](#_Toc43667820)

[1. PacketBuff 21](#_Toc43667821)

[2. PacketLayer 21](#_Toc43667822)

[3. PacketInfo 22](#_Toc43667823)

[4. SnifferClass 23](#_Toc43667824)

[III. Tầng Data Access 28](#_Toc43667825)

[1. NPF và NDIS 29](#_Toc43667826)

[2. Cấu trúc cơ bản của NPF 30](#_Toc43667827)

[KẾT QUẢ THỰC HIỆN (DEMO) 32](#_Toc43667828)

[TÀI LIỆU THAM KHẢO 37](#_Toc43667829)

[MỤC LỤC 38](#_Toc43667830)

1. Tham khảo thêm tại <https://biot.com/capstats/bpf.html> [↑](#footnote-ref-2)
2. Xem thêm tại <https://www.sciencedirect.com/topics/computer-science/berkeley-packet-filter> [↑](#footnote-ref-3)