IPK - Packet Sniffer

1.0.0

Generated by Doxygen 1.9.1

Chapter 1

Main Page

1.1 Introduction

This project is an implementation of a small and simple packet sniffer, implemented in C#. The packet sniffer application `ipk-sniffer` was build using *sharppacap* library and supports filtering a few different protocol and frame types, which will be listed bellow.

1.2 Build

This application is build using dotnet. However we also include a simple `Makefile` to make the build process of the application itself even simpler on linux. Therefore to build the application on linux, simply run:

```
make
```

To build the application on other platforms, which do not support make, use `dotnet release`.

1.3 Usage

1.3.1 Execution

After building, the application is used as follows.

```
./ipk-sniffer [-i interface | –interface interface] {-p port [–tcp|-t] [–udp|-u]} [–arp] [–icmp4] [–icmp6] [–igmp] [–mld] {-n num}
```

Where: `-i` eth0 (just one interface to sniff) or `--interface`. If this parameter is not specified (and any other parameters as well), or if only -i/–interface is specified without a value (and any other parameters are unspecified), a list of active interfaces is printed (additional information beyond the interface list is welcome but not required).
`-t or --tcp` (will display TCP segments and is optionally complemented by -p functionality).
`-u or --udp` (will display UDP datagrams and is optionally complemented by-p functionality).
`-p port_number` (extends previous two parameters to filter TCP/UDP based on port number; if this parameter is not present, then no filtering by port number occurs; if the parameter is given, the given port can occur in both the source and destination part of TCP/UDP headers).
`--icmp4` (will display only ICMPv4 packets).
`--icmp6` (will display only ICMPv6 echo request/response).
`--arp` (will display only ARP frames).
`--ndp` (will display only ICMPv6 NDP packets).
`--igmp` (will display only IGMP packets).
`--mld` (will display only MLD packets).
Unless protocols are explicitly specified, all (i.e., all content, regardless of protocol) are considered for printing.
`-n packet_count` (specifies the number of packets to display, i.e., the "time" the program runs; if not specified, consider displaying only one packet, i.e., as if -n 1)

∗∗ This information can also be displayed by using –help∗∗ Upon exit, either organically or using SIGINT, the application complies with standard bash exit codes [1]

1.3.2 Output

Non-printable characters are replaced with period. Output format:

> timestamp: time src MAC: MAC address with : as separator dst MAC: MAC address with : as separator frame length: length src IP: IP address if any (support v4 but also v6 representation according to RFC5952) dst IP: IP address if any (support v4 but also v6 representation according to RFC5952) src port: port number if any dst port: port number if any byte_offset: byte_offset_hexa byte_offset_ASCII

whereby:

- time is in RFC 3339 format
- length is in bytes

1.4 Implementation

In this section, we describe some of the implementation details and specifics.

1.5 Dependecies

The project uses these NuGet packages also included in ∗.csproj∗ file.

- PackageReference Include="CommandLineParser" Version="2.9.1"
- PackageReference Include="PacketDotNet" Version="1.4.7"
- PackageReference Include="SharpPcap" Version="6.2.5"

1.6 Design and Implementation

The application was designed using Singleton design pattern for class Sniffer, which handles the biggest part of it. It builds on the sharppacap library for packet capturing. Application operates in a way which we think allows for fast processing of incoming packets, as the main thread only enqueues incoming packets into a blocking queue and then the processing thread takes these packets one by one. Therefore the actual packet capturing is not slowed down by expensive operations, such as IO and packet parsing.

1.6.1 Device and packet filtering

Firstly, the device specified by interface parameter is opened and then filtering parameters are considered for building the final filter. The final filter is an "OR" of the separate filters, described bellow.

TCP and UDP filters:

> (TCP) (TCP port port_number)

Or

> (UDP) (UDP port port_number)

[2]

ICMPv4, ICMPv6 and ARP filters:

> (icmp) (icmp6) (arp)

NDP filter:

> (icmp6[icmp6type] = icmp6-neighborsolicit or icmp6[icmp6type] = icmp6-routersolicit or icmp6[icmp6type] = icmp6-routeradvert or icmp6[icmp6type] = icmp6-neighboradvert or icmp6[icmp6type] = icmp6-redirect)

[4]

MLD filter:

> (icmp6[icmp6type] = icmp6-multicastlistenerquery or icmp6[icmp6type] = icmp6-multicastlistenerreportv1 or icmp6[icmp6type] = icmp6-multicastlistenerreportv2 or icmp6[icmp6type] = icmp6-multicastlistenerdone)

[3]

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 Options Class Reference

Argument parser class

Static Public Member Functions

- static void HandleParseErrors ()

  Handle errors which may occur while parsing command line arguments
- static void CheckArguments (Options? arguments)

  Check correctness of command line arguments

Properties

- string? Interface `[get, set]`

  Interface
- int PacketLimit `[get, set]`

  Packet Limit
- bool Tcp `[get, set]`

  TCP
- bool Udp `[get, set]`

  UDP
- int Port `[get, set]`

  Port
- bool Icmp4 `[get, set]`

  ICMPv4
- bool Icmp6 `[get, set]`

  ICMPv6
- bool Arp `[get, set]`

  ARP
- bool Ndp `[get, set]`

  NDP
- bool Igmp `[get, set]`

  IGMP
- bool Mld `[get, set]`

  MLD

3.1.1 Detailed Description

Argument parser class

3.1.2 Member Function Documentation

3.1.2.1 CheckArguments()

```
static void Options.CheckArguments (
            Options? arguments )  [static]
```

Check correctness of command line arguments

Parameters

arguments	

3.1.2.2 HandleParseErrors()

```
static void Options.HandleParseErrors ( )  [static]
```

Handle errors which may occur while parsing command line arguments

3.1.3 Property Documentation

3.1.3.1 Arp

```
bool Options.Arp  [get], [set]
```

ARP

3.1.3.2 Icmp4

```
bool Options.Icmp4  [get], [set]
```

ICMPv4

3.1.3.3 Icmp6

```
bool Options.Icmp6  [get], [set]
```

ICMPv6

3.1.3.4 Igmp

```
bool Options.Igmp  [get], [set]
```

IGMP

3.1.3.5 Interface

```
string?  Options.Interface  [get], [set]
```

Interface

3.1.3.6 Mld

```
bool Options.Mld  [get], [set]
```

MLD

3.1.3.7 Ndp

```
bool Options.Ndp  [get], [set]
```

NDP

3.1.3.8 PacketLimit

```
int Options.PacketLimit  [get], [set]
```

Packet Limit

3.1.3.9 Port

```
int Options.Port  [get], [set]
```

Port

3.1.3.10 Tcp

```
bool Options.Tcp  [get], [set]
```

TCP

3.1.3.11 Udp

```
bool Options.Udp  [get], [set]
```

UDP

The documentation for this class was generated from the following file:

- ipk-sniffer/Program.cs

3.2 PacketInfoParsed Class Reference

Stores parsed packet information

Public Member Functions

- override string ToString ()
  All attributes converted to string containing the required wireshark packet hexdump format

Properties

- string TimeStamp = "" `[get, set]`
  Packet time stamp
- string SrcMac = "" `[get, set]`
  Source MAC address
- string DstMac = "" `[get, set]`
  Destination MAC address
- string FrameLenght = "" `[get, set]`
  Frame length in bytes
- string SrcIp = "" `[get, set]`
  Source IP address
- string DstIp = "" `[get, set]`
  Destination IP address
- string SrcPort = "" `[get, set]`
  Source port
- string DstPort = "" `[get, set]`
  Destination port
- string ByteOffset = "" `[get, set]`
  Hex and ascii dump

3.2.1 Detailed Description

Stores parsed packet information

3.2.2 Member Function Documentation

3.2.2.1 ToString()

```
override string PacketInfoParsed.ToString ( )
```

All attributes converted to string containing the required wireshark packet hexdump format

Returns

String representation of internal attributes

3.2.3 Property Documentation

3.2.3.1 ByteOffset

```
string PacketInfoParsed.ByteOffset = ""  [get], [set]
```

Hex and ascii dump

3.2.3.2 DstIp

```
string PacketInfoParsed.DstIp = ""  [get], [set]
```

Destination IP address

3.2.3.3 DstMac

```
string PacketInfoParsed.DstMac = ""  [get], [set]
```

Destination MAC address

3.2.3.4 DstPort

```
string PacketInfoParsed.DstPort = ""  [get], [set]
```

Destination port

3.2.3.5 FrameLenght

```
string PacketInfoParsed.FrameLenght = ""  [get], [set]
```

Frame length in bytes

3.2.3.6 SrcIp

```
string PacketInfoParsed.SrcIp = ""  [get], [set]
```

Source IP address

3.2.3.7 SrcMac

```
string PacketInfoParsed.SrcMac = ""  [get], [set]
```

Source MAC address

3.2.3.8 SrcPort

```
string PacketInfoParsed.SrcPort = ""  [get], [set]
```

Source port

3.2.3.9 TimeStamp

```
string PacketInfoParsed.TimeStamp = ""  [get], [set]
```

Packet time stamp

The documentation for this class was generated from the following file:

- ipk-sniffer/PacketParser.cs

3.3 Sniffer Class Reference

Sniffer class, packet handler designed as a singleton

Public Member Functions

- Sniffer (Options arguments)

  Create a Sniffer object
- void Initialize ()

  Initialize packet sniffer Open device and set PacketArrival handler
- void CapturePackets ()

  Capture packets and process them in a separate thread
- void Filter (Options arguments)

  Set filter for PacketSniffer

Static Public Member Functions

- static void PrintDevices ()

  List all available devices

3.3.1 Detailed Description

Sniffer class, packet handler designed as a singleton

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Sniffer()

```
Sniffer.Sniffer (
                Options arguments )
```

Create a Sniffer object

Parameters

arguments	Command line arguments which influence the way how packet Sniffer is created

3.3.3 Member Function Documentation

3.3.3.1 CapturePackets()

```
void Sniffer.CapturePackets ( )
```

Capture packets and process them in a separate thread

Starts capturing packets on device then start PacketProcessing thread and waits for the thread to be ended and joined back, meaning that correct number of packets was handled. After that, capturing is stopped

3.3.3.2 Filter()

```
void Sniffer.Filter (
            Options arguments )
```

Set filter for PacketSniffer

Parameters

arguments	Command line arguments containing wanted filters

3.3.3.3 Initialize()

```
void Sniffer.Initialize ( )
```

Initialize packet sniffer Open device and set PacketArrival handler

3.3.3.4 PrintDevices()

```
static void Sniffer.PrintDevices ( )  [static]
```

List all available devices

The documentation for this class was generated from the following file:

- ipk-sniffer/Sniffer.cs

Bibliography

[1] Appendix e. exit codes with special meanings. 2

[2] Capture filters. 3

[3] S. Deering, W. Fenner, and B. Haberman. Multicast listener discovery (mld) for ipv6, Oct 1999. 3

[4] T. Narten, E. Nordmark, W. Simpson, and H. Soliman. Neighbor discovery for ip version 6 (ipv6), Sep 1970. 3

Index