Communication Networks

LAN Measurement

User Guide

Németh Krisztián

BME TMIT

March 2024

v1.1

In this User Guide, we have summarized the theoretical knowledge and some practical information required for measurement. It is not absolutely necessary to read it before starting the measurement, but during the measurement it is worth stopping from time to time and turning the pages of this document to the relevant parts.

Content

1 Theoretical background

1.1 Ethernet frames

Ethernet is an umbrella term for a family of local area network technologies. We call it a "local area" network because its size is typically no larger than an apartment, small company, or a department of a larger company. At the university, even machines belonging to one department are often connected by dividing them into several Ethernet networks. In TCP/IP networks, it is logically located below the IP layer.

Ethernet is defined by the IEEE 802.3 standards group. In the data connection layer, transmission takes place in packet-switched form. Its basic unit is the Ethernet frame. The frame is preceded by a preamble and a start frame delimiter (SFD), which are used for the purpose of synchronization. The frame starts with different header fields, which are shown in Fig Fig 1.(1 octet = 1 byte = 8 bits). The header is followed by the payload, which usually embeds protocol data units of additional layers (L3, e.g. IP). This is followed by a 32-bit checksum (cyclic redundancy check, CRC), which serves to check the integrity of the frame. Finally, the so-called frame gap (interframe/interpacket gap, IFG/IPG) follows.

Layer	Preamblem	Start Frame Delimi-ter (SFD)	MAC dest. addr.	MAC source addr.	802.1q tag (optional)	EtherType or length	Payload	CRC	IFG/IPG
	7 octet	1 octet	6 octet	6 octet	(4 octet)	2 octet	46-1500 octet	4 octet	12 octet
L2: Ethernet frame			← 64–1522 octet →						
L1: Ethernet packet and IPG	← 72–1530 octet →								← 12 octet →

Fig 1.. 802.3 Ethernet frame structure (from: wikipedia)

1.1.1 MAC addresses

MAC (Medium Access Control) addresses are the identifiers used in Ethernet for individual endpoints (i.e. typically their network cards). MAC addresses are 6 octets (48 bits) long.

Although these 6 octets together form an indivisible MAC address, it can still be logically divided into two parts. The first three octets are called Organizationally Unique Identifier (OUI), and the other three octets are network interface controller (NIC) specific unique serial numbers. Certain bit positions of the address have a special meaning:

- Bit 0 (LSB) of the first octet of the OUI:
  - 0: unicast address
  - 1: multicast address

- Bit 1 of the first octet of the OUI:
    - 0: globally unique (OUI enforced)
    - 1: locally administered

The all-1 (ff:ff:ff:ff:ff:ff) address is called broadcast address that all nodes in the broadcast domain receive. Packets addressed to a unicast address are received by the interface if they are addressed to its physical address. To catch frames sent to other destinations, the so-called the connector must be in promiscuous mode. The latter means that the interface also processes frames that are not intended for it.

Since most hardware manufacturers have a registered OUI (or more), there are many databases that can be used to find out from a MAC address which manufacturer the device itself is from.

1.1.2 802.1q tag

The 802.1q tag, also known as the VLAN tag is essential for creating virtual local networks. VLANs are discussed in detail in the next chapter.

1.1.3 EtherType

This two-octet field defines the type of protocol data unit embedded in the frame. During the measurements, we will meet the so-called Ethernet II, also known as DIX (DEC+Intel+Xerox) frames. Typical EtherType values:

- 0x0800: IPv4 data
- 0x0806: ARP frame
- 0x86dd: IPv6 data

These frames can be saved on the PC with traffic capture applications (e.g. Wireshark, tcpdump, etc.), but we will only see the fields from the destination MAC address to the end of the payload, but not the preamble, SFD and CRC !

1.2 ARP

We send IP packets on a TCP/IP network, in the header of which we specify, among other things, the source and destination IP addresses. For example, if I want to download a page from a web server on the local network, the DNS tells my machine the corresponding IP address, and the browser sends an http request to it in an IP packet.

Yes, but my computer (phone) has an Ethernet (and/or WiFi) card that can send and receive Ethernet (WLAN) frames. It is fine to embed the IP packet in such a frame, even the source address field of the Ethernet (WLAN) frame is given (the MAC address of my network card), but what should be the destination address? I mean the target Ethernet (WiFi) address?

Obviously, the MAC address of the network card of the web server. OK, but how do we learn that? This is what the ARP protocol is for, and the functions that implement it in the operating system. First of all, we may have communicated with this IP address recently, and then we may still remember its MAC address. So there is a table in our machine with the assignment of recently used IP addresses and MAC addresses. You can check this with the `arp -a` command on the Windows command line, or with the `arp` command on Linux.

But what if the IP address you are looking for is not in this table? The Address Resolution Protocol (ARP) was created for this problem. The protocol is not new, it was specified in RFC 826 in 1982, although it has been expanded several times since then. In principle, ARP works with different upper and lower layer protocols, but

we only use IP (upper) and Ethernet/WLAN (lower) protocols in this measurement, and in fact only with IP and Ethernet.

At first glance, the operation of ARP is quite simple: our machine sends a request to the broadcast MAC address ff:ff:ff:ff:ff:ff, in which it describes that it is looking for the MAC address corresponding to the given IP address. This is processed by all devices on the local network. The one who recognizes this IP address replies, now only to the requester, giving its own MAC address. After that, communication in IP packets embedded in Ethernet frames can begin. And, of course, the obtained MAC address - IP address association is included by the machine's operating system in the previously mentioned ARP table, so that it does not have to be polled again before each packet.

1.3 VLANs

Broadcast messages are received by all endpoints within an Ethernet network. Such broadcast messages are also Address Resolution Protocol (ARP) messages. Traditionally, computers that are physically close to each other are plugged into the same Ethernet switch. If in a company the machines of one department (e.g. finance) are close to each other and the machines belonging to another department (e.g. HR) are located on another floor, it is easy to create a separate Ethernet network for each of them. This is beneficial in several ways:

- manageability: the administrator of each department is responsible for his own network
- scalability: broadcast messages remain within the department, they do not flood the entire company
- security: the possibility of abuse within the LAN is also limited

However, if, in another example, the computers of two (or more) departments are mixed on the same floor, it would still be advisable to separate them. Two Ethernet switches can be an unnecessary investment if the capacity of one is sufficient, but isolation is not possible with one.

The concept of virtual local networks (Virtual LAN, VLAN) was born to solve this problem. Using this, you only need one Ethernet switch, and you can set, for example, which port belongs to which VLAN. The VLANs are separated from each other by the Ethernet switch, and no Ethernet message broadcasting or any other Ethernet-level (layer 2) communication is possible between them.

In addition, it will also be possible to connect the Ethernet switches of, say, two floors in such a way that the machines of the two departments are connected to both, and to logically divide the whole into two VLANs. This also requires that the Ethernet frames in the communication between the two switches be marked as to which VLAN they belong to. This is what the so-called VLAN tag (the 802.1q tag in the header) is for.

The way tags are used is detailed in the IEEE 802.1q standard, sometimes abbreviated as .1q. Placing the tag (tagging) means that a tag with a given ID is placed on an incoming/outgoing packet on an interface, that is, the optional 4 octets are inserted into the Ethernet header. The first two are the protocol identifier (tag protocol identifier, TPID) with the value 0x8100, and the second two are tag control information (TCI): 3 bits of priority code point (PCP), 1 bit of DEI (drop eligible indicator) and 12 bits of VLAN ID (VID). A physical port can be an untagged or tagged member of a given VLAN. In untagged mode, even though the interface is part of the VLAN's broadcast range, the frames distributed through it are not tagged. For this, it is important to remember that a port can only be a member of one VLAN in untagged mode, the membership of other VLANs must already be in tagged mode.

Trunk connections typically carry such tagged frames, except for those without such a tag: these belong to the so-called native VLAN. Before the packet reaches that port operating in untagged mode, the tag must be removed. This operation is *untagging.*

So each VLAN behaves as an independent broadcast domain. Each such VLAN is assigned an identifier (tag) from the range 0...4095, which is practically unique at the site level. We implement this logical topology on the L2 and L3 devices of the infrastructure:

- (typically) on backbone connections, through which the traffic of several VLANs must be routed, we always distribute tagged frames (even in the case that currently only the traffic of a single VLAN is transmitted, because in this way we can easily assign additional VLANs to the backbone connection in the future)
- on the access ports on which end-point devices are connected to a VLAN, Ethernet frames are forwarded without tags

The Ethernet frames to be forwarded from the access ports to the backbone port are tagged, and the outgoing ones are untagged. On the backbone connections (uplink ports), we also practically perform VLAN filtering: we discard the frames that do not belong to any of the VLANs assigned to the backbone connection, and forward the others as usual.

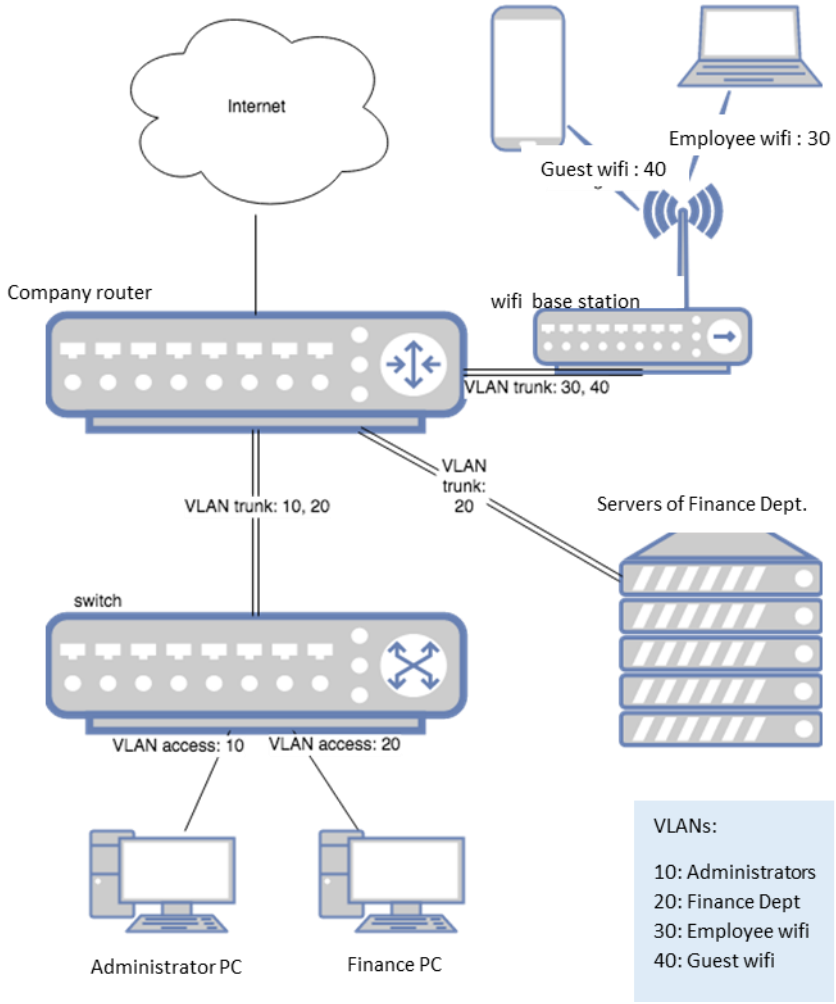


Fig. 2. A typical small business network

As shown in Figure 2, individual VLANs can be formed arbitrarily through a network device that supports 802.1q. Thus, e.g. financial VLAN 20 is accessible from the financial workstation through the given access port of the switch, but the administrative machine located on the other access port 10 of this switch cannot communicate on the given subnet.

It is important to note that VLANs are independent broadcasting domains, between which communication is only possible using IP routing, this is called inter-VLAN routing. If an Ethernet switch also supports L3 mode, this function can be activated on it. Thus, although the nodes located in different VLANs cannot communicate with each other with MAC address, with the help of inter-VLAN routing in the network layer (IP addressing), it is possible to exchange messages between them via the given switch.

1.4 Firewall

The typical task of a firewall is to monitor network traffic passing through a certain layer and to limit or allow traffic according to a specific set of rules. The most common firewall in network devices is a packet-level (typically up to layer 4) processing. Additional functions also occur, such as network address translation (NAT) and port address translation (PAT). There are also application-level firewalls, but their application on communication paths is not trivial due to the increasingly common encryption nowadays, and serious heuristics are required for their operation.

A typical packet-level firewall has 3 basic rule systems, or chains in technical terms: forward, input, output. The first applies to the traffic passing through it, the second to the traffic sent to the specific device, and the third to the traffic originating from the device. Each chain can be assigned a default policy: allow or discard, depending on what happens to packets that do not meet any of the chain's rules.

Our rules can then be defined in many ways in relation to the given chains, e.g. destination/source address, protocol, port number, interface, connection status, and many other methods related to network protocols. In the rule, we must also specify an action that occurs when the match is made, e.g. accept, drop, etc.

When the firewall is operating, the packets to be processed will be judged according to the valid rule chains during packet forwarding: if a rule matches, the associated operation will be executed. If not and it can be continued (the rule did not say stop), matching continues with the next rule. And after the rules, the default policy comes into force.

For example, we can set the firewall to allow all packets coming from a specific IP address range (source address) or sent to a specific port (e.g. http: port 80) into a domain, but discard all other packets.

1.5 Mikrotik RouterOS

RouteOS is Mikrotik's self-developed, Linux-based but closed-source code firmware, primarily for use on its own RouterBoard-based platform (switches, routers, base stations). Its advantage is uniformity (same firmware for all hardware), and the long support, i.e. that modern firmware can also be installed on old hardware. When purchasing the device, we also receive the firmware, but certain functions require a license fee, e.g. installation on x86 PC as well.

RouterOS can be configured most conveniently with the free graphical application WinBox (the "Practical Guide" introduces you to its use), but it can also be done from a web browser (WebFig) and from the command line (via ssh or GUIs) by experienced operators.

The CAPsMAN (Controlled Access Point system Manager) function helps to create a corporate network consisting of several APs: APs can be classified as CAPs (controlled access points), since then they only perform AP tasks (access control, user identification) and these settings (e.g. RADIUS server, WPA/WPA2 keys) can be configured centrally, or even centrally can also be updated.

1.5.1 hAP ac^2

The #1 port (Internet/POE In) of the 1+4-wire Gigabit Ethernet ports of the router used for the measurement is assigned to the WAN connection (but can also be configured as a LAN port). Wired ports are identified by RouterOS as *ether1* to *ether5*.

Our router also has two radio interfaces: one for serving the 802.11b/g/n standard, called *wlan1*. The 802.11ac standard is operated by the interface called *wlan2*. All interfaces can be disabled individually.

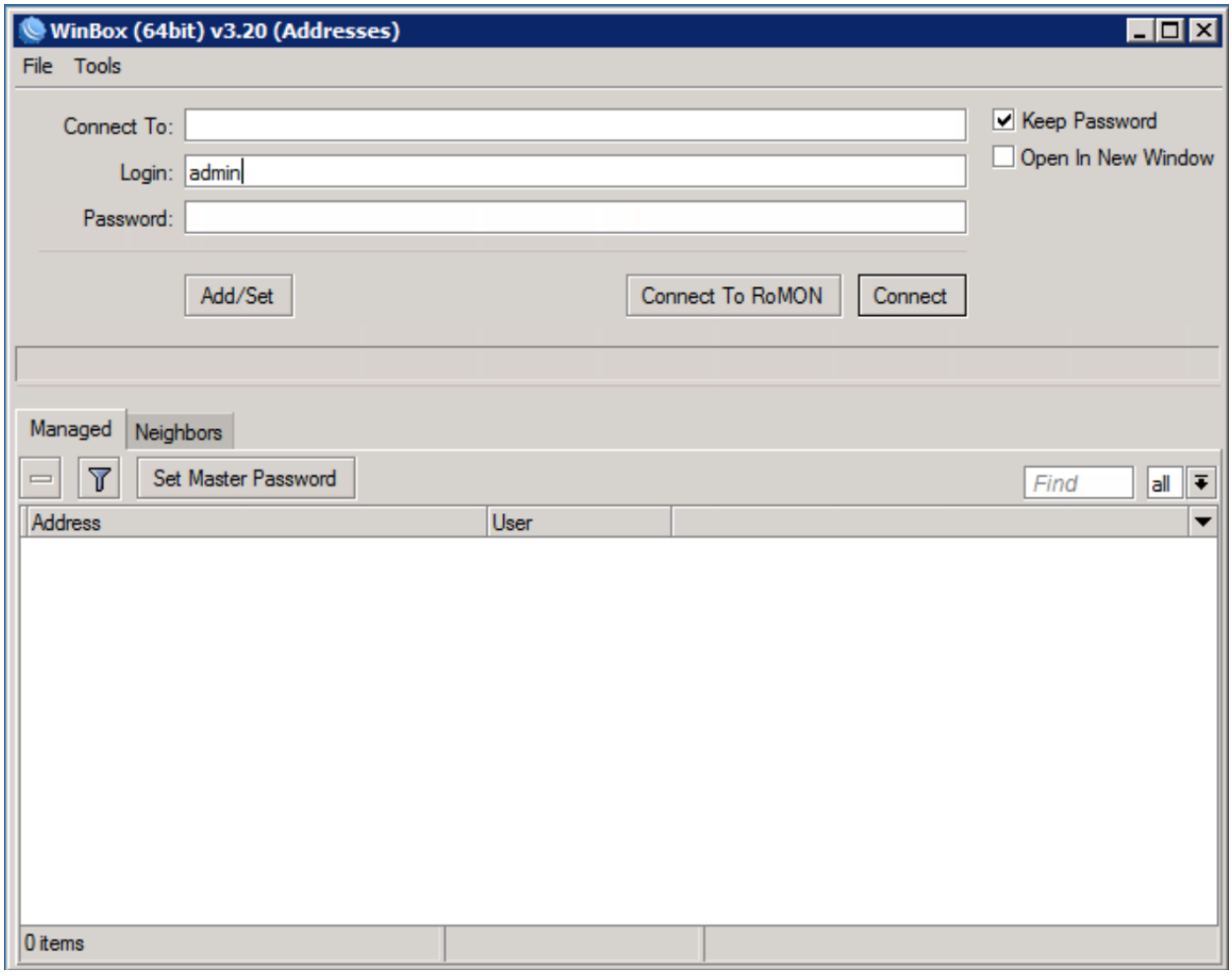
2 Practical info

2.1 How to use WinBox

2.1.1 Connection, Main menu

To start the program, look for the following icon: 

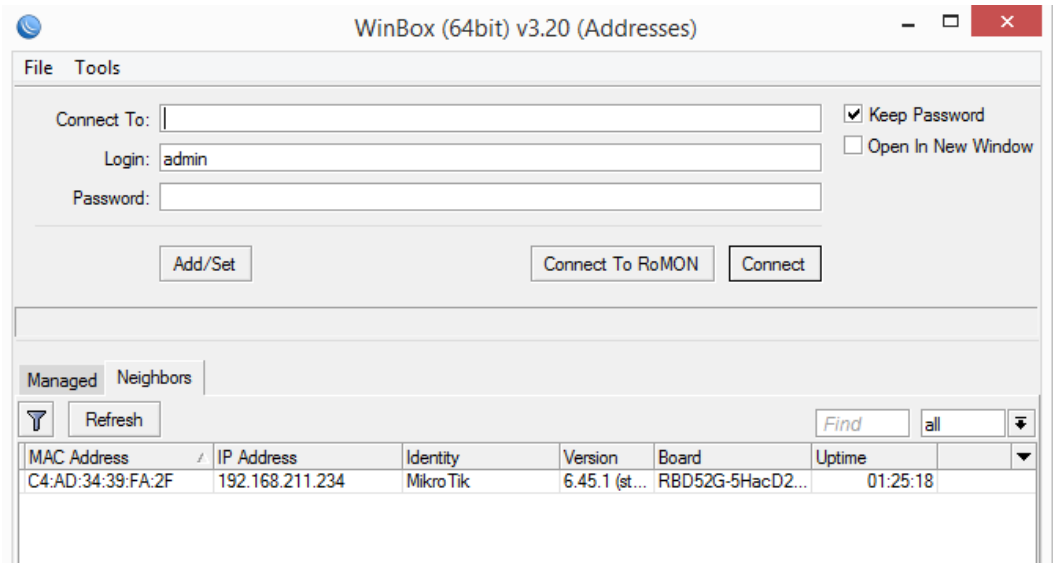
After starting, you can see the following window:



On the *Managed* tab, in the *Connect To* field, write the IP address or MAC address of the device you want to manage, as well as the administrator login name (*admin* by default) and the corresponding password (empty by default). WinBox remembers the address of the last successfully connected device, or we can add more here.

As above (*Managed* tab), we can give the address of the device we want to manage. It is also possible to choose from the automatically recognized surrounding devices, for this there is the *Neighbors* tab.

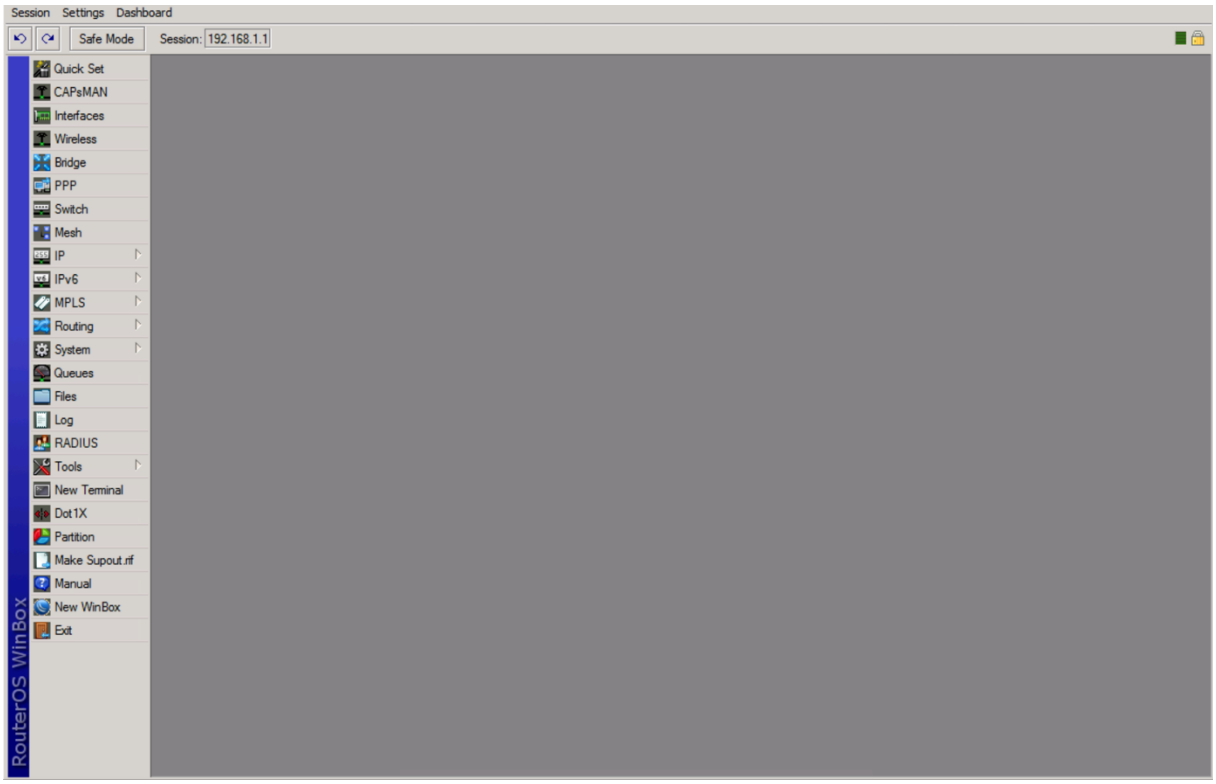
So, on the *Neighbors* tab, you can find a list of detected devices running RouterOS:



The *Identity* field can help the identification (if we have already given a unique name) or the MAC and IP address. Make sure you always connect to the device you really want to configure!

To connect, click the *Connect* button! (RoMON – Router Management Overlay Network – is a centralized management layer, we do not work with it during the measurements, so we use *Connect* instead of the *Connect to RoMON* button!)

After connecting, we can see a picture similar to this:



Although the configuration settings became valid immediately, activating the so-called Safe Mode can be beneficial. If enabled, WinBox will ask for confirmation to finalize when exiting. If this is not done, it will undo all settings made in the session. In this way, we can prevent ourselves from being excluded, or so that the device has to be reset.

The dashboard bar below the menu bar can be extended with a few interesting measuring tools that can be accessed via the Dashboard menu: date, time, CPU load, memory usage and uptime (time elapsed since power-on).

The left sidebar of the main window is the Main Menu. Its most important points:

- Quick Set: quick setting "wizard", for quick setting of the most common functions (e.g. typical home Wi-Fi router setup)
- CAPsMAN: functions required for building a centralized Wi-Fi network
- Interfaces: interfaces, VLANs
- Wireless: radios, wireless profiles
- Bridge: creating a network bridge
- PPP: creation of point-to-point connections (e.g. dial-up, PPPoE)
- Switch: configuration of the built-in network switch
- Mesh: network of wireless devices (decentralized)
- IP, IPv6: IP settings
- Routing: route selection
- System: device administration (e.g. exact time, saving licenses, settings, etc.)
- Dot1X: endpoint authentication settings
- Log: log of system messages
- Tools: useful tools (ping, traffic generator, etc.)
- New Terminal: command line window (e.g. for scripting)

The main menu items with a right arrow next to them also have a second level, the leaf items open a window each. WinBox is a multi-window interface, which makes it extremely convenient to review the lot of functions. The windows can have tabs or a toolbar that typically looks like this:

- +/- button: add/delete (depending on the context), for some functions + also covers a drop-down menu (if there is a down arrow next to the sign)
- checkmark/cross: enable/disable selected elements. Attention, its use is effective immediately!
- yellow note (comment): adding a comment to the item. It will usually appear above the item in the list, unless "Inline Comments" is enabled in the Settings menu.
- filter: if it can be interpreted in the context, it can be used to search for certain elements in the list

2.1.2 Interfaces menu

The most important tabs:

- *Interface* tab: list of individual interfaces, including radio, virtual, VLAN, bridge, etc. Here they can also be disabled (see checkmark/cross toolbar button), and the most important statistics can also be displayed
- *Interface List* tab: the interfaces can be grouped so that they can be referenced together (e.g. in a firewall, etc.), by default the LAN and WAN groups are here
- *Ethernet* tab: the wired ethernet connectors
- *VLAN* tab: defined VLANs, when adding a new one, we must specify at least the VLAN ID and the name of the parent interface

2.1.3 Bridge menu

- • *Bridge* tab: list of bridge connections
- • *Ports*: the interface-bridge assignments
- • *VLANs*: here, VLANs created in the Interfaces menu can be assigned to bridges, and tagging (adding tag) and untagging (tag removal) rules can be specified. For those that have already been created, we can also see the current tagging and boring mechanism

…

2.1.4 IP menu

- *ARP*: ARP table: we can see which MAC-IP associations the device knows. In justified cases, we can also add new entries
- *Addresses*: list of IP addresses: we can assign IP addresses to interfaces, including VLANs, which can be used as gateways (or management addresses)
- *DHCP Client*: if you want to request an IP address on an interface, you can give it here. Typically, part of the default configuration is to request an address via DHCP for the interface serving as the WAN connection.
- *DHCP Server*: if we want to share an IP address on a VLAN (on the bridge according to the default configuration), we can define a DHCP server here:
    - *DHCP* tab: server/interface list
    - *Networks* tab: the DHCP server must know the address, netmask and gateway of the given network (typically the address specified in *Addresses* tab), you can enter them here
    - *Leases*: here you can list which IP addresses we have assigned to which MAC address, or how long they are valid. We can add or delete static registrations.
    - *Options/Option Sets*: if we want to advertise special DHCP parameters, e.g. boot server address, etc., then we can add them here. The DNS server name and gateway do not need to be entered separately here.
- *DNS*: DNS cache settings of the device
- *Firewall*: firewall settings
    - *Filter Rules*: management of filter rules; # indicates order, ::: indicates comments. In addition to the condition fields, we can also see the quantity (*Bytes* column) and number (*Packets* column) of matching packets, this can help in troubleshooting
    - *NAT*: address/port translation rules
    - *Connections*: connection registry table

…

- *Pool*: enter IP address ranges, from which the DHCP server can allocate addresses. You can give a name to the pool, you can say the range from which addresses can be allocated (e.g. 192.168.88.2-192.168.88.10), and which should be the next pool if it has already run out
- *Routes*: route table, gateways. We can also add static ones if needed. When defining an interface or a VLAN, the route is automatically entered here.

2.1.5 System menu

- *Clock*: system time and date, even with NTP synchronization
- *Identity*: the hostname of the device, we can also change it here
- *Packages*: download additional packages
- *Password*: changing of admin password
- *Reset Configuration*: restoring factory settings
- *Reboot*: restart. It's rarely needed.

…

2.2 Wireshark: what is it and how to use

Wireshark is an open source protocol analysation software. Network traffic can be captured and saved (in pcap/pcapng format), which can be opened, analyzed and played back later (e.g. with tcpreplay). In addition to displaying the binary content of the packages, the software is capable of decoding known protocols and displaying headers. In addition, in the case of certain protocols (e.g. TCP), it is capable of tracking packet streams, i.e. also displaying related packets together.

We call it packet capture, when we catch and save the packets with wireshark arriving on the given network interface (advanced people can even redirect them, using so-called named pipes). The interception takes place in practically promiscuous mode, which allows saving not only the packets intended for the intercepting node, but also everything that the interface "sees". This can be useful, for example, in the case of a wireless network connection or port mirroring.

By default, the capture can be done with the GUI, but also from the console, using the command-line application called dumpcap.

2.2.1 Capture filter

During packet capture, it can be useful, especially in the case of an aggregated connection, if we do not save all packets, only those that we are definitely interested in during the examination. In such cases, with the so-called capture filter, i.e. a capture-level filtering expression, we can specify what Wireshark should receive/save.

The most frequently used expressions:

- filtering for IP address: "`host` x.x.x.x" or "`host` domainname "
- filtering for subnet: "`net` network/maskbits" or "`net` network `mask` netmask"
- filtering to source: "`src`" prefix
- filtering to destination: "d`st`" prefix
- filtering to protocol: "`proto` x", where x can be "udp", "tcp", "icmp", "arp" etc.; "`proto`" can be omitted for the first three
- filtering to port: "`port` x", where x can either be IANA service name, e.g. "`port` domain" or "`port` 53"
- filtering to port range, e.g.: "tcp[0:2]>1500 and tcp[0:2]<1550", i.e. the first two bytes of the TCP header (source)
- newer libpcaps (above 0.9.1): "tcp portrange x-y"
- special:
    - "broadcast", "multicast"

The elements of the expression can be connected with the logical operators *and* or *or*, they can be bracketed, and the *not* operator can also be used.

2.2.2 Display filter

You can also filter in the series of packages that have already been caught. This filtering only affects the display, not the content of the saved file. If the expression matches it, it will be displayed, otherwise not. The syntax of the expression is slightly different from the capture filter:

- logical expressions: matching protocol fields with operators, e.g. ip.src, tcp.window_size
- operators: ==, eq, !=, contains, matches, !, e.g. ip.src==192.168.0.0/16, tcp.port eq 25

- sequence of expressions separated by logical switches: &&, ||, and, or, e.g. http.request.uri matches "gl=se$"
- see Wireshark display filter reference for more information: https://wiki.wireshark.org/DisplayFilters

2.2.3 Protocol analysis

The detailed protocol information of the selected package can be viewed in the package list, by opening the headers of each layer layer by layer.

It may happen that the software does not automatically recognize which protocols are embedded, for example RTP, which is embedded in UDP, but does not have a standard port number or a unique header. In this case, by selecting the package, you can explicitly say which protocol we are dealing with: Decode as...

Typical web browsing involves opening multiple parallel TCP connections from time to time. Wireshark can extract individual TCP streams and assigns a stream identifier (tcp.stream) to each of them.

- Statistics → Conversations: protokoll+forrás/cél IP cím/port alapján felderíti a folyamokat és megjeleníti. Itt látható, hogy mettől meddig zajlottak a folyamok, mennyi adat mozgott rajtuk
- Statistics → I/O Graph: átviteli grafikont tudunk rajzoltatni egy vagy több kiválasztott csomagsorozatra, amit display filterrel tudunk meghatározni
- Telephony → RTP Streams: hasonló, mint a Conversations, de az RTP folyamnak bizonyos tulajdonságait is méri, pl. csomagvesztés, jitter
- Statistics Conversations: detects and displays flows based on protocol+source/destination IP address/port. Here you can see how long the streams have been running since then, and how much data has moved on them
- Statistics I/O Graph: we can draw a transmission graph for one or more selected series of packets, which can be defined with a display filter
- Telephony RTP Streams: similar to Conversations, but also measures certain properties of the RTP stream, e.g. packet loss, jitter

2.2.4 Introduction to using Wireshark

This chapter is much more detailed than what is actually needed for measurement!

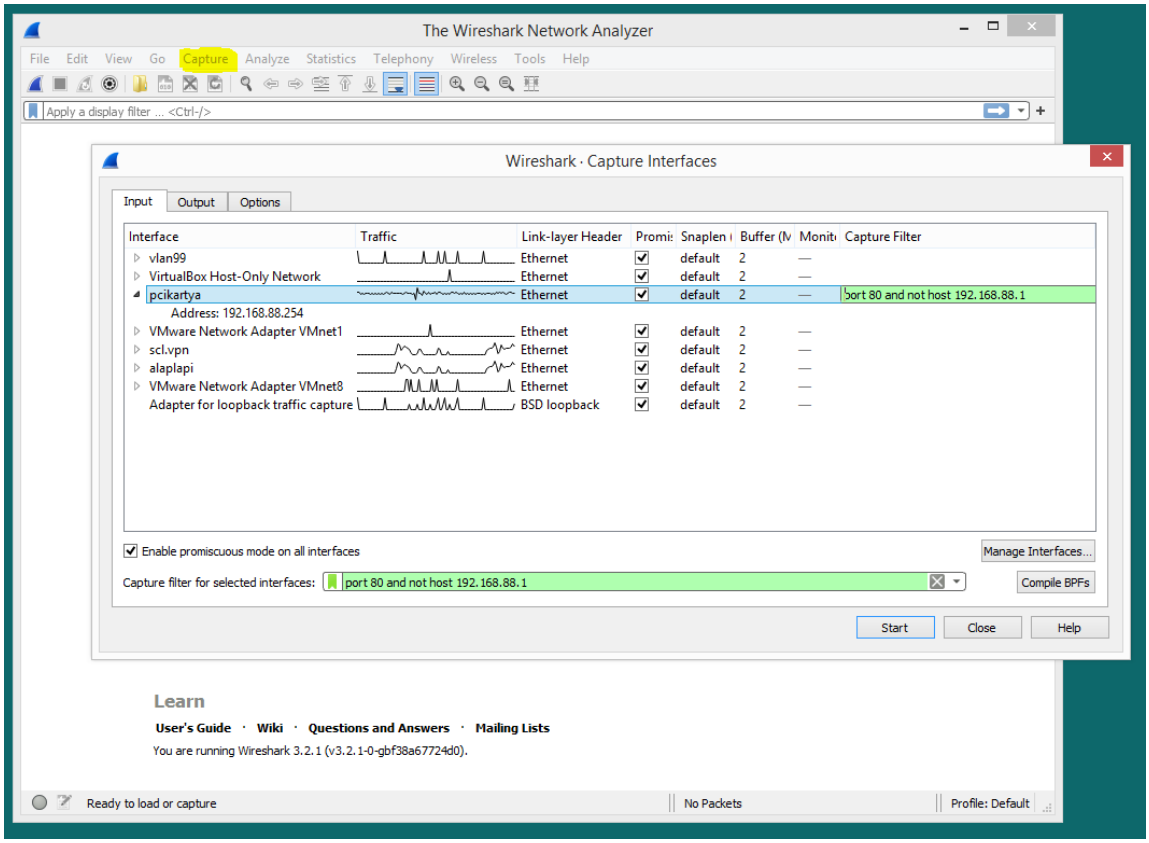
To start it Wireshark, look for its icon which is similar to this one: 

The opening screen contains a list of interfaces from which we can capture packets with a given authorization level, as well as a list of recently opened pcap files (files containing captured packets).

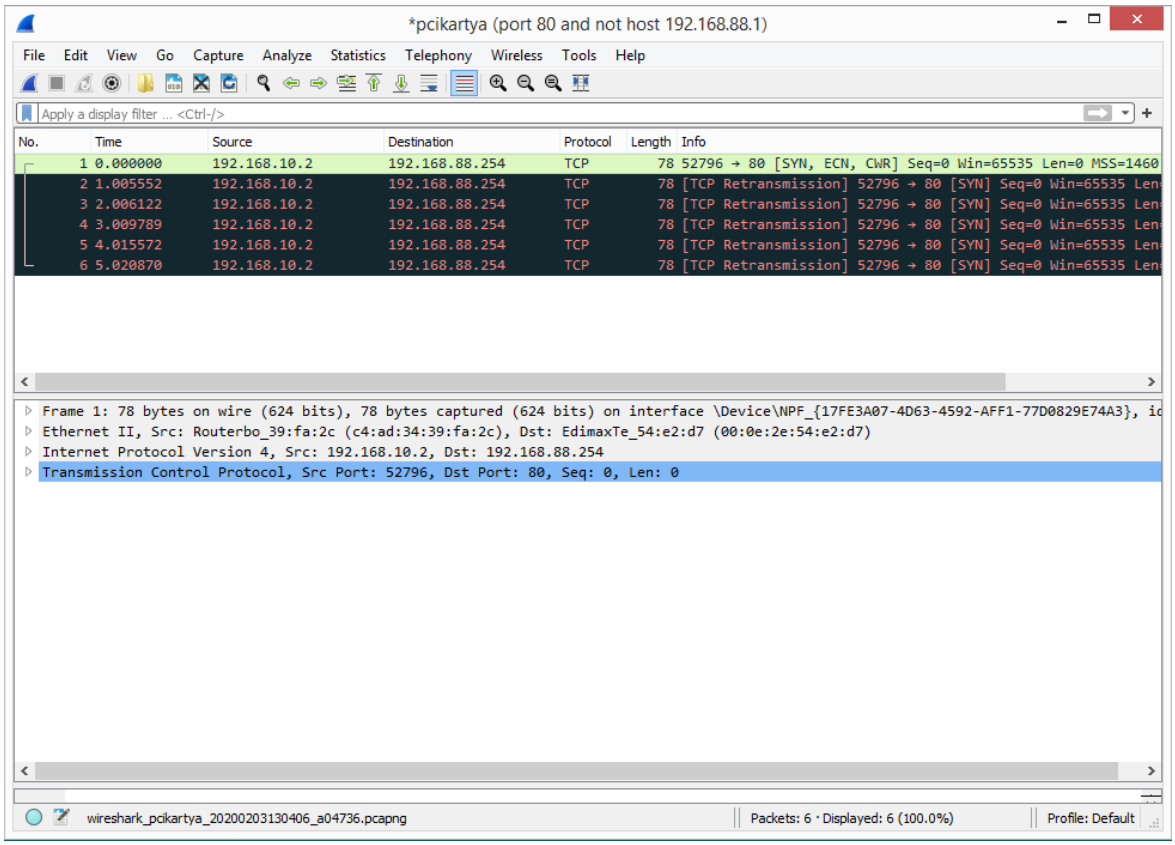
Start packet capture on a given interface with a simple capture filter1:

Capture → Options, find the interface in question (its IP address and name can also help) and optionally enter a rule in the Capture Filter column, then Start

[1] Here we can say that Wireshark should not receive all the packages, only certain ones. Later, with the help of the "display filter", we can select the ones we want to display from among the already caught packets.



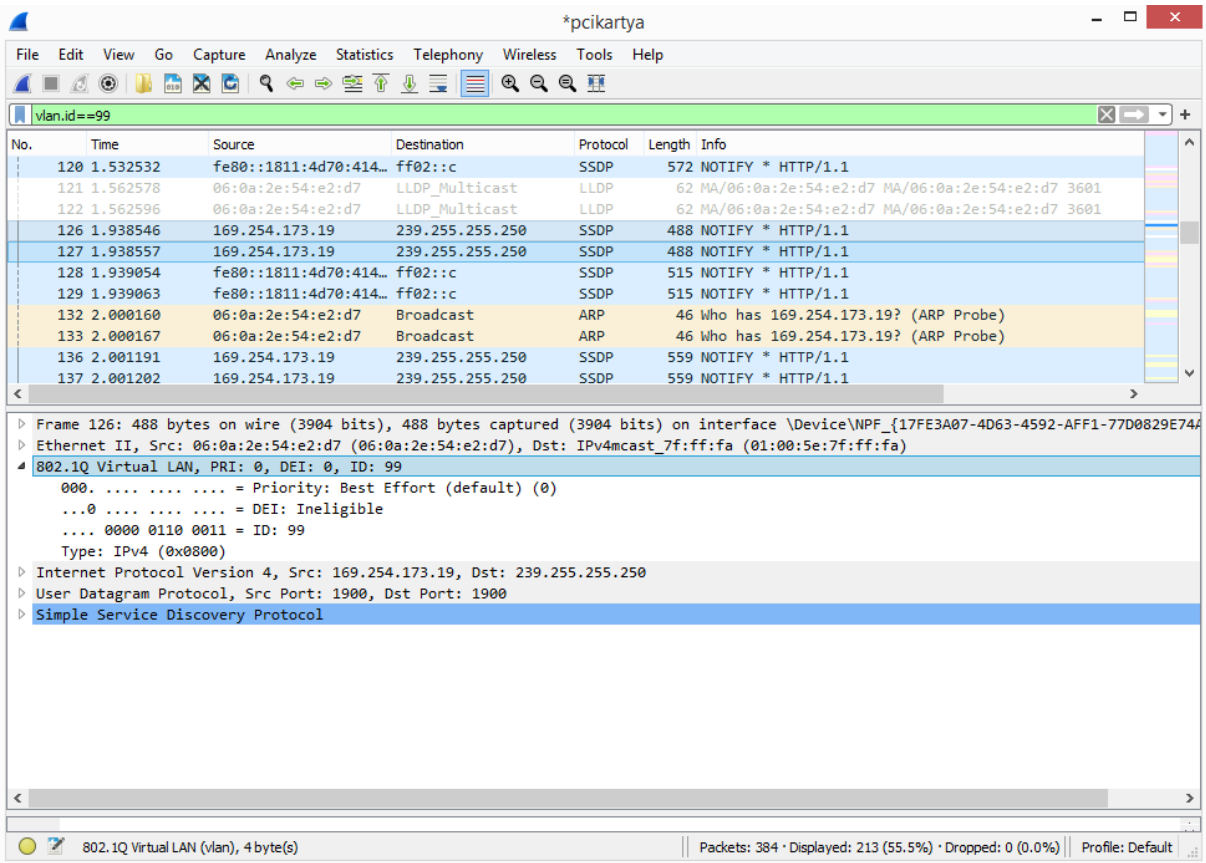
After starting the capture, we get to the screen with the following arrangement (as a result of the packet capture rule above). We see that a TCP connection is being established from the address 192.168.10.2 (packet #1, TCP-SYN). Normally, we should see a TCP-ACK packet in response to this. We do not see this in this example, only SYN packets arriving again at certain intervals. That is, this TCP connection could not be established.



Below the menu and toolbar, we find a text input bar similar to the address bar of browsers. This is used to specify the display filter expression. If empty, we will see all captured packets.

Below this we will see the series of packages. Sorted by catch by default. The package list is constantly expanding, until we stop the capture with the stop button. The package selected in the list can be viewed in detail in the block with drop-down menus below the package list. Here, the header of each recognized protocol will appear in the form of a drop-down menu, where the header fields related to the protocol can be viewed. The embedding can of course be viewed by moving inwards.

In the figure below, we can see the use of the display filter: with the expression written in the address line, only packets with the given property (now with VLAN ID 99) are displayed.



2.2.5 Remote packet capture

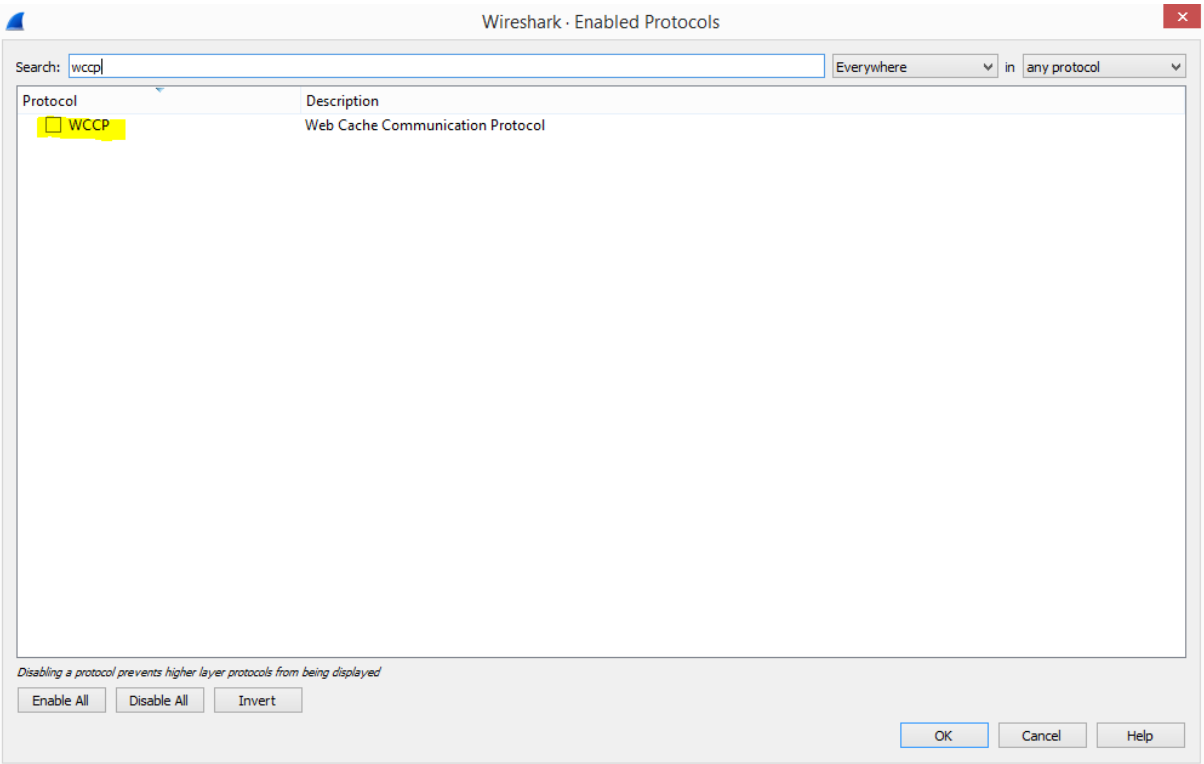
It can be a useful function if we can isolate the traffic appearing on the router in some way and direct it to a remote node or even save it. This function can be configured in *Tools* → *Packet Sniffer* of the RouterOS.

The sniffer can be switched on and off with the *Start / Stop* button. Changing the settings is only possible when it is switched off. If you want to save the traffic to the router, you should plug a suitable USB drive into the USB port, then enter the path and file name under *File Name*. (The path to the file system attached from the USB drive can be found in the *Files* menupoint.)

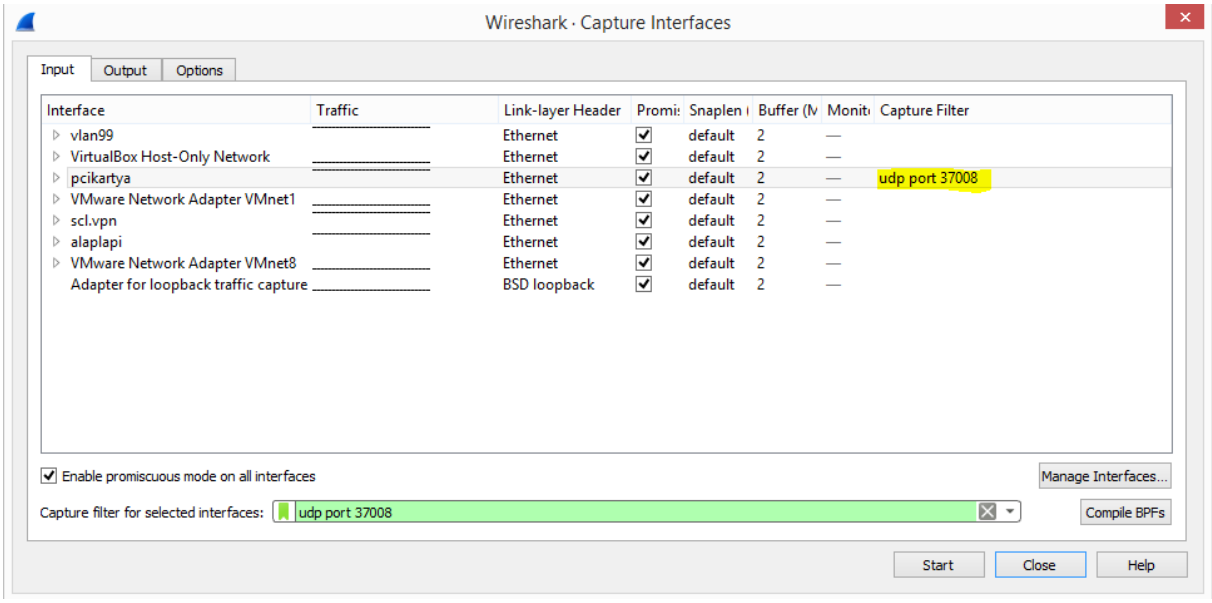
The settings for remote capture can be found in the *Streaming* tab. You can enable the transmission of the captured traffic by ticking *Streaming Enabled*. For the *Server*, we must enter the IP address of the node. If we want to filter the traffic (e.g. catching certain types of packets, etc.), then we have to check the *Filter Stream* and we can set various filtering options on the *Filter* tab. It can be interesting to capture traffic passing through a given interface, or capture according to a given IP protocol or port number

After making the settings (*Apply*), you can start streaming with *Start*. The sniffer will embed the traffic in UDP packets destined for port 37008.

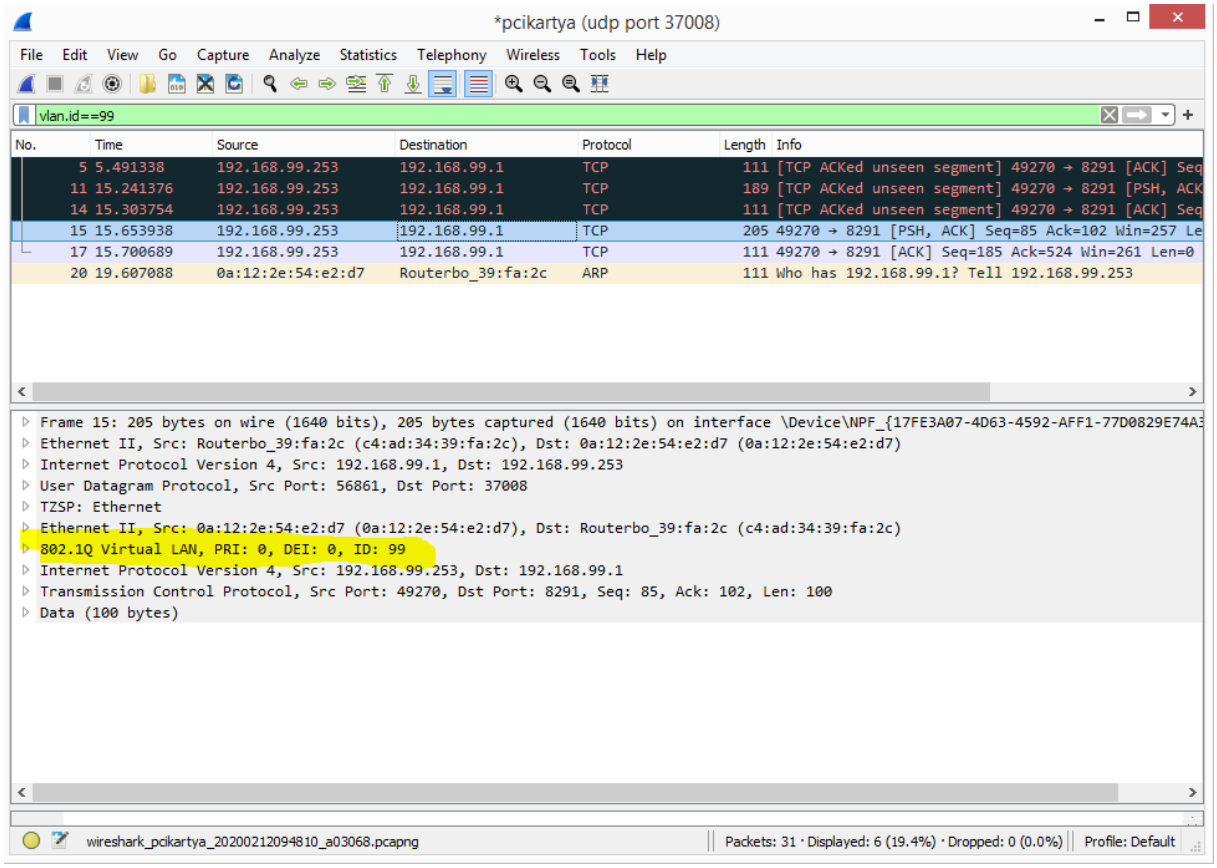
In order to be able to capture and view these packets with Wireshark, we must first disable WCCP protocol recognition, which we must find in the Wireshark *Analyze → Enabled Protocols* menu and remove the checkmark:



Then, start the capture so that only the packets arriving at the above port number are received:



In the UDP packets caught in this way, Ethernet frames will be embedded, containing additional protocols, as they were caught by the sniffer running on the router. In the screenshot below, we can also see a packet containing a VLAN tag:


*pcikartya (udp port 37008)
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
vlan.id==99
No. Time Source Destination Protocol Length Info
5 5.491338 192.168.99.253 192.168.99.1 TCP 111 [TCP ACKed unseen segment] 49270 → 8291 [ACK] Seq
11 15.241376 192.168.99.253 192.168.99.1 TCP 189 [TCP ACKed unseen segment] 49270 → 8291 [PSH, ACK
14 15.303754 192.168.99.253 192.168.99.1 TCP 111 [TCP ACKed unseen segment] 49270 → 8291 [ACK] Seq
15 15.653938 192.168.99.253 192.168.99.1 TCP 205 49270 → 8291 [PSH, ACK] Seq=85 Ack=102 Win=257 Le
17 15.700689 192.168.99.253 192.168.99.1 TCP 111 49270 → 8291 [ACK] Seq=185 Ack=524 Win=261 Len=0
20 19.607088 0a:12:2e:54:e2:d7 Routerbo_39:fa:2c ARP 111 Who has 192.168.99.1? Tell 192.168.99.253
Frame 15: 205 bytes on wire (1640 bits), 205 bytes captured (1640 bits) on interface \Device\NPF_{17FE3A07-4D63-4592-AFF1-77D0829E74A3
Ethernet II, Src: Routerbo_39:fa:2c (c4:ad:34:39:fa:2c), Dst: 0a:12:2e:54:e2:d7 (0a:12:2e:54:e2:d7)
Internet Protocol Version 4, Src: 192.168.99.1, Dst: 192.168.99.253
User Datagram Protocol, Src Port: 56861, Dst Port: 37008
TZSP: Ethernet
Ethernet II, Src: 0a:12:2e:54:e2:d7 (0a:12:2e:54:e2:d7), Dst: Routerbo_39:fa:2c (c4:ad:34:39:fa:2c)
802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 99
Internet Protocol Version 4, Src: 192.168.99.253, Dst: 192.168.99.1
Transmission Control Protocol, Src Port: 49270, Dst Port: 8291, Seq: 85, Ack: 102, Len: 100
Data (100 bytes)
wireshark_pcikartya_20200212094810_a03068.pcapng
Packets: 31 · Displayed: 6 (19.4%) · Dropped: 0 (0.0%)
Profile: Default