NAME

wireshark − Interactively dump and analyze network traffic

SYNOPSIS

wireshark [**−i** <capture interface>|−] [**−f** <capture filter>] [**−Y** <display filter>] [**−w** <outfile>] [**options**] [<infile>]

DESCRIPTION

Wireshark is a GUI network protocol analyzer. It lets you interactively browse packet data from a live network or from a previously saved capture file. **Wireshark**'s native capture file formats are **pcapng** format and **pcap** format; it can read and write both formats.. **pcap** format is also the format used by **tcpdump** and various other tools; **tcpdump**, when using newer verions of the **libpcap** library, can also read some pcapng files, and, on newer versions of macOS, can read all pcapng files and can write them as well.

Wireshark can also read / import the following file formats:

- Oracle (previously Sun) **snoop** and **atmsnoop** captures
- Finisar (previously Shomiti) **Surveyor** captures
- Microsoft **Network Monitor** captures
- Novell **LANalyzer** captures
- AIX’s **iptrace** captures
- Cinco Networks **NetXRay** captures
- NETSCOUT (previously Network Associates/Network General) Windows−based **Sniffer** captures
- Network General/Network Associates DOS−based **Sniffer** captures (compressed or uncompressed)
- LiveAction (previously WildPackets/Savvius) ***Peek**/**EtherHelp**/**PacketGrabber** captures
- **RADCOM**'s WAN/LAN analyzer captures
- Viavi (previously Network Instruments) **Observer** captures
- **Lucent/Ascend** router debug output
- captures from HP−UX **nettl**
- **Toshiba’s** ISDN routers dump output
- the output from **i4btrace** from the ISDN4BSD project
- traces from the **EyeSDN** USB S0
- the **IPLog** format output from the Cisco Secure Intrusion Detection System
- **pppd logs** (pppdump format)
- the output from VMS’s **TCPIPtrace**/**TCPtrace**/**UCX$TRACE** utilities
- the text output from the **DBS Etherwatch** VMS utility
- Visual Networks' **Visual UpTime** traffic capture
- the output from **CoSine** L2 debug
- the output from InfoVista (previously Accellent) **5View** LAN agents
- Endace Measurement Systems' ERF format captures
- Linux Bluez Bluetooth stack **hcidump −w** traces
- Catapult DCT2000 .out files
- Gammu generated text output from Nokia DCT3 phones in Netmonitor mode
- IBM Series (OS/400) Comm traces (ASCII & UNICODE)

- Juniper Netscreen snoop files
- Symbian OS btsnoop files
- TamoSoft CommView files
- Tektronix K12xx 32bit .rf5 format files
- Tektronix K12 text file format captures
- Apple PacketLogger files
- Captures from Aethra Telecommunications' PC108 software for their test instruments
- Citrix NetScaler Trace files
- Android Logcat binary and text format logs
- Colasoft Capsa and PacketBuilder captures
- Micropross mplog files
- Unigraf DPA−400 DisplayPort AUX channel monitor traces
- 802.15.4 traces from Daintree’s Sensor Network Analyzer
- MPEG−2 Transport Streams as defined in ISO/IEC 13818−1
- Log files from the *candump* utility
- Logs from the BUSMASTER tool
- Ixia IxVeriWave raw captures
- Rabbit Labs CAM Inspector files
- *systemd* journal files
- 3GPP TS 32.423 trace files

There is no need to tell **Wireshark** what type of file you are reading; it will determine the file type by itself. **Wireshark** is also capable of reading any of these file formats if they are compressed using gzip. **Wireshark** recognizes this directly from the file; the '.gz' extension is not required for this purpose.

Like other protocol analyzers, **Wireshark**'s main window shows 3 views of a packet. It shows a summary line, briefly describing what the packet is. A packet details display is shown, allowing you to drill down to exact protocol or field that you interested in. Finally, a hex dump shows you exactly what the packet looks like when it goes over the wire.

In addition, **Wireshark** has some features that make it unique. It can assemble all the packets in a TCP conversation and show you the ASCII (or EBCDIC, or hex) data in that conversation. Display filters in **Wireshark** are very powerful; more fields are filterable in **Wireshark** than in other protocol analyzers, and the syntax you can use to create your filters is richer. As **Wireshark** progresses, expect more and more protocol fields to be allowed in display filters.

Packet capturing is performed with the pcap library. The capture filter syntax follows the rules of the pcap library. This syntax is different from the display filter syntax.

Compressed file support uses (and therefore requires) the zlib library. If the zlib library is not present, **Wireshark** will compile, but will be unable to read compressed files.

The pathname of a capture file to be read can be specified with the **−r** option or can be specified as a command−line argument.

OPTIONS

Most users will want to start **Wireshark** without options and configure it from the menus instead. Those users may just skip this section.

−a|−−autostop <capture autostop condition>

Specify a criterion that specifies when **Wireshark** is to stop writing to a capture file. The criterion is of the form *test:value*, where *test* is one of:

duration:*value* Stop writing to a capture file after *value* seconds have elapsed. Floating point values (e.g. 0.5) are allowed.

files:*value* Stop writing to capture files after *value* number of files were written.

filesize:*value* Stop writing to a capture file after it reaches a size of *value* kB. If this option is used together with the −b option, Wireshark will stop writing to the current capture file and switch to the next one if filesize is reached. Note that the filesize is limited to a maximum value of 2 GiB.

packets:*value* Stop writing to a capture file after it contains *value* packets. Same as **−c**<capture packet count>.

−b|−−ring−buffer <capture ring buffer option>

Cause **Wireshark** to run in "multiple files" mode. In "multiple files" mode, **Wireshark** will write to several capture files. When the first capture file fills up, **Wireshark** will switch writing to the next file and so on.

The created filenames are based on the filename given with the **−w** flag, the number of the file and on the creation date and time, e.g. outfile_00001_20220714120117.pcap, outfile_00002_20220714120523.pcap, ...

With the *files* option it's also possible to form a "ring buffer". This will fill up new files until the number of files specified, at which point **Wireshark** will discard the data in the first file and start writing to that file and so on. If the *files* option is not set, new files filled up until one of the capture stop conditions match (or until the disk is full).

The criterion is of the form *key:value*, where *key* is one of:

duration:*value* switch to the next file after *value* seconds have elapsed, even if the current file is not completely filled up. Floating point values (e.g. 0.5) are allowed.

files:*value* begin again with the first file after *value* number of files were written (form a ring buffer). This value must be less than 100000. Caution should be used when using large numbers of files: some filesystems do not handle many files in a single directory well. The **files** criterion requires one of the other criteria to be specified to control when to go to the next file. It should be noted that each **−b** parameter takes exactly one criterion; to specify two criteria, each must be preceded by the **−b** option.

filesize:*value* switch to the next file after it reaches a size of *value* kB. Note that the filesize is limited to a maximum value of 2 GiB.

interval:*value* switch to the next file when the time is an exact multiple of *value* seconds.

packets:*value* switch to the next file after it contains *value* packets.

Example: **−b filesize:1000 −b files:5** results in a ring buffer of five files of size one megabyte each.

−B|−−buffer−size <capture buffer size>

Set capture buffer size (in MiB, default is 2 MiB). This is used by the capture driver to buffer packet data until that data can be written to disk. If you encounter packet drops while capturing, try to increase this size. Note that, while **Wireshark** attempts to set the buffer size to 2 MiB by default, and can be told to set it to a larger value, the system or interface on which you're capturing might silently limit the capture buffer size to a lower value or raise it to a higher value.

This is available on UNIX systems with libpcap 1.0.0 or later and on Windows. It is not available on UNIX systems with earlier versions of libpcap.

This option can occur multiple times. If used before the first occurrence of the **−i** option, it sets the default capture buffer size. If used after an **−i** option, it sets the capture buffer size for the interface specified by the last **−i** option occurring before this option. If the capture buffer size is not set specifically, the default capture buffer size is used instead.

−c <capture packet count>

Set the maximum number of packets to read when capturing live data. Same as **−a packets:**<capture packet count>.

−C <configuration profile>

Start with the given configuration profile.

−−capture−comment <comment>

When performing a capture file from the command line, with the **−k** flag, add a capture comment to the output file, if supported by the capture format.

This option may be specified multiple times. Note that Wireshark currently only displays the first comment of a capture file.

−d <layer type>==<selector>,<decode−as protocol>

Like Wireshark's **Decode As...** feature, this lets you specify how a layer type should be dissected. If the layer type in question (for example, **tcp.port** or **udp.port** for a TCP or UDP port number) has the specified selector value, packets should be dissected as the specified protocol.

Example: **−d tcp.port==8888,http** will decode any traffic running over TCP port 8888 as HTTP.

See the tshark(1) manual page for more examples.

−D|−−list−interfaces

Print a list of the interfaces on which **Wireshark** can capture, and exit. For each network interface, a number and an interface name, possibly followed by a text description of the interface, is printed. The interface name or the number can be supplied to the **−i** flag to specify an interface on which to capture.

This can be useful on systems that don't have a command to list them (UNIX systems lacking **ifconfig −a** or Linux systems lacking **ip link show**). The number can be useful on Windows systems, where the interface name might be a long name or a GUID.

Note that "can capture" means that **Wireshark** was able to open that device to do a live capture; if, on your system, a program doing a network capture must be run from an account with special privileges (for example, as root), then, if **Wireshark** is run with the **−D** flag and is not run from such an account, it will not list any interfaces.

−−display <X display to use>

Specifies the X display to use. A hostname and screen (otherhost:0.0) or just a screen (:0.0) can be specified. This option is not available under Windows.

−−disable−protocol <proto_name>

Disable dissection of proto_name.

−−disable−heuristic <short_name>

Disable dissection of heuristic protocol.

−−enable−protocol <proto_name>

Enable dissection of proto_name.

−−enable−heuristic <short_name>

Enable dissection of heuristic protocol.

−f <capture filter>

Set the capture filter expression.

This option can occur multiple times. If used before the first occurrence of the **−i** option, it sets the default capture filter expression. If used after an **−i** option, it sets the capture filter expression for the interface specified by the last **−i** option occurring before this option. If the capture filter expression is not set specifically, the default capture filter expression is used if provided.

Pre−defined capture filter names, as shown in the GUI menu item Capture→Capture Filters, can be used by prefixing the argument with "predef:". Example: **−f "predef:MyPredefinedHostOnlyFilter"**

−−fullscreen

Start Wireshark in full screen mode (kiosk mode). To exit from fullscreen mode, open the View menu and select the Full Screen option. Alternatively, press the F11 key (or Ctrl + Cmd + F for macOS).

−g <packet number>

After reading in a capture file using the **−r** flag, go to the given *packet number*.

−h|−−help

Print the version number and options and exit.

−H

Hide the capture info dialog during live packet capture.

−i|−−interface <capture interface>|−

Set the name of the network interface or pipe to use for live packet capture.

Network interface names should match one of the names listed in "**wireshark −D**" (described above); a number, as reported by "**wireshark −D**", can also be used. If you’re using UNIX, "**netstat −i**", "**ifconfig −a**" or "**ip link**" might also work to list interface names, although not all versions of UNIX support the **−a** option to **ifconfig**.

If no interface is specified, **Wireshark** searches the list of interfaces, choosing the first non−loopback interface if there are any non−loopback interfaces, and choosing the first loopback interface if there are no non−loopback interfaces. If there are no interfaces at all, **Wireshark** reports an error and doesn’t start the capture.

Pipe names should be either the name of a FIFO (named pipe) or "−" to read data from the standard input. On Windows systems, pipe names must be of the form "\\pipe\.*pipename*". Data read from pipes must be in standard pcapng or pcap format. Pcapng data must have the same endianness as the capturing host.

"TCP@<host>:<port>" causes **Wireshark** to attempt to connect to the specified port on the specified host and read pcapng or pcap data.

This option can occur multiple times. When capturing from multiple interfaces, the capture file will be saved in pcapng format.

−I|−−monitor−mode

Put the interface in "monitor mode"; this is supported only on IEEE 802.11 Wi−Fi interfaces, and supported only on some operating systems.

Note that in monitor mode the adapter might disassociate from the network with which it’s associated, so that you will not be able to use any wireless networks with that adapter. This could prevent accessing files on a network server, or resolving host names or network addresses, if you are capturing in monitor mode and are not connected to another network with another adapter.

This option can occur multiple times. If used before the first occurrence of the **−i** option, it enables the monitor mode for all interfaces. If used after an **−i** option, it enables the monitor mode for the interface specified by the last **−i** option occurring before this option.

−j

Use after **−J** to change the behavior when no exact match is found for the filter. With this option select the first packet before.

−J <jump filter>

After reading in a capture file using the **−r** flag, jump to the packet matching the filter (display filter syntax). If no exact match is found the first packet after that is selected.

−k

Start the capture session immediately. If the **−i** flag was specified, the capture uses the specified interface. Otherwise, **Wireshark** searches the list of interfaces, choosing the first non−loopback interface if there are any non−loopback interfaces, and choosing the first loopback interface if there are no non−loopback interfaces; if there are no interfaces, **Wireshark** reports an error and doesn’t start the capture.

−K <keytab>

Load kerberos crypto keys from the specified keytab file. This option can be used multiple times to load keys from several files.

Example: **−K krb5.keytab**

−l

Turn on automatic scrolling if the packet display is being updated automatically as packets arrive during a capture (as specified by the **−S** flag).

−L|−−list−data−link−types

List the data link types supported by the interface and exit.

−−list−time−stamp−types

List time stamp types supported for the interface. If no time stamp type can be set, no time stamp types are listed.

−n

Disable network object name resolution (such as hostname, TCP and UDP port names), the **−N** flag might override this one.

−N <name resolving flags>

Turn on name resolving only for particular types of addresses and port numbers, with name resolving for other types of addresses and port numbers turned off. This flag overrides **−n** if both **−N** and **−n** are present. If both **−N** and **−n** flags are not present, all name resolutions are turned on.

The argument is a string that may contain the letters:

m to enable MAC address resolution

n to enable network address resolution

N to enable using external resolvers (e.g., DNS) for network address resolution

t to enable transport−layer port number resolution

d to enable resolution from captured DNS packets

v to enable VLAN IDs to names resolution

−o <preference/recent setting>

Set a preference or recent value, overriding the default value and any value read from a preference/recent file. The argument to the flag is a string of the form *prefname:value*, where *prefname* is the name of the preference/recent value (which is the same name that would appear in the preference/recent file), and *value* is the value to which it should be set. Since **Ethereal** 0.10.12, the recent settings replaces the formerly used −B, −P and −T flags to manipulate the GUI dimensions.

If *prefname* is "uat", you can override settings in various user access tables using the form uat*:*uat filename*:*uat record*. *uat filename* must be the name of a UAT file, e.g. *user_dlts*. *uat_record* must be in the form of a valid record for that file, including quotes. For instance, to specify a user DLT from the command line, you would use

```
−o "uat:user_dlts:\"User 0 (DLT=147)\",\"cops\",\"0\",\"\",\"0\",\"\""
```

−p|−−no−promiscuous−mode

Don't put the interface into promiscuous mode. Note that the interface might be in promiscuous mode for some other reason; hence, **−p** cannot be used to ensure that the only traffic that is captured is traffic sent to or from the machine on which **Wireshark** is running, broadcast traffic, and multicast traffic to addresses received by that machine.

This option can occur multiple times. If used before the first occurrence of the **−i** option, no interface will be put into the promiscuous mode. If used after an **−i** option, the interface specified by the last **−i** option occurring before this option will not be put into the promiscuous mode.

−P <path setting>

Special path settings usually detected automatically. This is used for special cases, e.g. starting Wireshark from a known location on an USB stick.

The criterion is of the form *key:path*, where *key* is one of:

persconf:*path* path of personal configuration files, like the preferences files.

persdata:*path* path of personal data files, it's the folder initially opened. After the very first initialization, the recent file will keep the folder last used.

−r|−−read−file <infile>

Read packet data from *infile*, can be any supported capture file format (including gzipped files). It's not possible to use named pipes or stdin here! To capture from a pipe or from stdin use **−i −**

−R|−−read−filter <read (display) filter>

When reading a capture file specified with the **−r** flag, causes the specified filter (which uses the syntax of display filters, rather than that of capture filters) to be applied to all packets read from the capture file; packets not matching the filter are discarded.

−s|−−snapshot−length <capture snaplen>

Set the default snapshot length to use when capturing live data. No more than *snaplen* bytes of each network packet will be read into memory, or saved to disk. A value of 0 specifies a snapshot length of 262144, so that the full packet is captured; this is the default.

This option can occur multiple times. If used before the first occurrence of the **−i** option, it sets the default snapshot length. If used after an **−i** option, it sets the snapshot length for the interface specified by the last **−i** option occurring before this option. If the snapshot length is not set specifically, the default snapshot length is used if provided.

−S

Automatically update the packet display as packets are coming in.

−t a|ad|adoy|d|dd|e|r|u|ud|udoy

Set the format of the packet timestamp displayed in the packet list window. The format can be one of:

a absolute: The absolute time, as local time in your time zone, is the actual time the packet was captured, with no date displayed

ad absolute with date: The absolute date, displayed as YYYY−MM−DD, and time, as local time in your time zone, is the actual time and date the packet was captured

adoy absolute with date using day of year: The absolute date, displayed as YYYY/DOY, and time, as local time in your time zone, is the actual time and date the packet was captured

d delta: The delta time is the time since the previous packet was captured

dd delta_displayed: The delta_displayed time is the time since the previous displayed packet was captured

e epoch: The time in seconds since epoch (Jan 1, 1970 00:00:00)

r relative: The relative time is the time elapsed between the first packet and the current packet

u UTC: The absolute time, as UTC, is the actual time the packet was captured, with no date displayed

ud UTC with date: The absolute date, displayed as YYYY−MM−DD, and time, as UTC, is the actual time and date the packet was captured

udoy UTC with date using day of year: The absolute date, displayed as YYYY/DOY, and time, as UTC, is the actual time and date the packet was captured

The default format is relative.

−−time−stamp−type <type>

Change the interface’s timestamp method. See −−list−time−stamp−types.

−u <s|hms>

Output format of seconds (def: s: seconds)

−v|−−version

Print the full version information and exit.

−w <outfile>

Set the default capture file name, or '−' for standard output.

−X <eXtension options>

Specify an option to be passed to an **Wireshark** module. The eXtension option is in the form *extension_key:value*, where *extension_key* can be:

lua_script:*lua_script_filename* tells **Wireshark** to load the given script in addition to the default Lua scripts.

lua_script*num*:*argument* tells **Wireshark** to pass the given argument to the lua script identified by 'num', which is the number indexed order of the 'lua_script' command. For example, if only one script was loaded with '−X lua_script:my.lua', then '−X lua_script1:foo' will pass the string 'foo' to the 'my.lua' script. If two scripts were loaded, such as '−X lua_script:my.lua' and '−X lua_script:other.lua' in that order, then a '−X lua_script2:bar' would pass the string 'bar' to the second lua script, namely 'other.lua'.

read_format:*file_format* tells **Wireshark** to use the given file format to read in the file (the file given in the **−r** command option).

stdin_descr:*description* tells **Wireshark** to use the given description when capturing from standard input (**−i** −).

−y|−−linktype <capture link type>

If a capture is started from the command line with **−k**, set the data link type to use while capturing packets. The values reported by **−L** are the values that can be used.

This option can occur multiple times. If used before the first occurrence of the **−i** option, it sets the default capture link type. If used after an **−i** option, it sets the capture link type for the interface specified by the last **−i** option occurring before this option. If the capture link type is not set specifically, the default capture link type is used if provided.

−Y|−−display−filter <displaY filter>

Start with the given display filter.

−z <statistics>

Get **Wireshark** to collect various types of statistics and display the result in a window that updates in semi−real time.

Some of the currently implemented statistics are:

−z help

Display all possible values for **−z**.

−z afp,srt[,*filter*]

Show Apple Filing Protocol service response time statistics.

−z conv,*type*[,*filter*]

Create a table that lists all conversations that could be seen in the capture. *type* specifies the conversation endpoint types for which we want to generate the statistics; currently the supported ones are:

```
"eth"   Ethernet addresses
"fc"    Fibre Channel addresses
"fddi"  FDDI addresses
"ip"    IPv4 addresses
"ipv6"  IPv6 addresses
"ipx"   IPX addresses
"tcp"   TCP/IP socket pairs   Both IPv4 and IPv6 are supported
"tr"    Token Ring addresses
"udp"   UDP/IP socket pairs   Both IPv4 and IPv6 are supported
```

If the optional *filter* is specified, only those packets that match the filter will be used in the calculations.

The table is presented with one line for each conversation and displays the number of packets/bytes in each direction as well as the total number of packets/bytes. By default, the table is sorted according to the total number of packets.

These tables can also be generated at runtime by selecting the appropriate conversation type from the menu "Tools/Statistics/Conversation List/".

−z dcerpc,srt,*name−or−uuid*,*major*.*minor*[,*filter*]

Collect call/reply SRT (Service Response Time) data for DCERPC interface *name* or *uuid*, version *major*.*minor*. Data collected is the number of calls for each procedure, MinSRT, MaxSRT and AvgSRT. Interface *name* and *uuid* are case−insensitive.

Example: **−z dcerpc,srt,12345778−1234−abcd−ef00−0123456789ac,1.0** will collect data for the CIFS SAMR Interface.

This option can be used multiple times on the command line.

If the optional *filter* is provided, the stats will only be calculated on those calls that match that filter.

Example: **−z dcerpc,srt,12345778−1234−abcd−ef00−0123456789ac,1.0,ip.addr==1.2.3.4** will collect SAMR SRT statistics for a specific host.

−z dhcp,stat[,*filter*]

Show DHCP (BOOTP) statistics.

−z expert

Show expert information.

−z fc,srt[,*filter*]

Collect call/reply SRT (Service Response Time) data for FC. Data collected is the number of calls for each Fibre Channel command, MinSRT, MaxSRT and AvgSRT.

Example: **−z fc,srt** will calculate the Service Response Time as the time delta between the First packet of the exchange and the Last packet of the exchange.

The data will be presented as separate tables for all normal FC commands, Only those commands that are seen in the capture will have its stats displayed.

This option can be used multiple times on the command line.

If the optional *filter* is provided, the stats will only be calculated on those calls that match that filter.

Example: **−z "fc,srt,fc.id==01.02.03"** will collect stats only for FC packets exchanged by the host at FC address 01.02.03 .

−z h225,counter[,*filter*]

Count ITU−T H.225 messages and their reasons. In the first column you get a list of H.225 messages and H.225 message reasons which occur in the current capture file. The number of occurrences of each message or reason is displayed in the second column.

Example: **−z h225,counter**

This option can be used multiple times on the command line.

If the optional *filter* is provided, the stats will only be calculated on those calls that match that filter.

Example: **−z "h225,counter,ip.addr==1.2.3.4"** will collect stats only for H.225 packets exchanged by the host at IP address 1.2.3.4 .

−z h225,srt[,*filter*]

Collect request/response SRT (Service Response Time) data for ITU−T H.225 RAS. Data collected is the number of calls of each ITU−T H.225 RAS Message Type, Minimum SRT, Maximum SRT, Average SRT, Minimum in Packet, and Maximum in Packet. You will also get the number of Open Requests (Unresponded Requests), Discarded Responses (Responses without matching request) and Duplicate Messages.

Example: **−z h225,srt**

This option can be used multiple times on the command line.

If the optional *filter* is provided, the stats will only be calculated on those calls that match that filter.

Example: **−z "h225,srt,ip.addr==1.2.3.4"** will collect stats only for ITU−T H.225 RAS packets exchanged by the host at IP address 1.2.3.4 .

−z io,stat

Collect packet/bytes statistics for the capture in intervals of 1 second. This option will open a window with up to 5 color−coded graphs where number−of−packets−per−second or number−of−bytes−per−second statistics can be calculated and displayed.

This option can be used multiple times on the command line.

This graph window can also be opened from the Analyze:Statistics:Traffic:IO−Stat menu item.

−z ldap,srt[,*filter*]

Collect call/reply SRT (Service Response Time) data for LDAP. Data collected is the number of calls for each implemented LDAP command, MinSRT, MaxSRT and AvgSRT.

Example: **−z ldap,srt** will calculate the Service Response Time as the time delta between the Request and the Response.

The data will be presented as separate tables for all implemented LDAP commands, Only those commands that are seen in the capture will have its stats displayed.

This option can be used multiple times on the command line.

If the optional *filter* is provided, the stats will only be calculated on those calls that match that filter.

Example: use **−z "ldap,srt,ip.addr==10.1.1.1"** will collect stats only for LDAP packets exchanged by the host at IP address 10.1.1.1 .

The only LDAP commands that are currently implemented and for which the stats will be available are: BIND SEARCH MODIFY ADD DELETE MODRDN COMPARE EXTENDED

−z megaco,srt[,*filter*]

Collect request/response SRT (Service Response Time) data for MEGACO. (This is similar to **−z smb,srt**). Data collected is the number of calls for each known MEGACO Command, Minimum SRT, Maximum SRT and Average SRT.

Example: **−z megaco,srt**

This option can be used multiple times on the command line.

If the optional *filter* is provided, the stats will only be calculated on those calls that match that filter.

Example: **−z "megaco,srt,ip.addr==1.2.3.4"** will collect stats only for MEGACO packets exchanged by the host at IP address 1.2.3.4 .

−z mgcp,srt[,*filter*]

Collect request/response SRT (Service Response Time) data for MGCP. (This is similar to **−z smb,srt**). Data collected is the number of calls for each known MGCP Type, Minimum SRT, Maximum SRT and Average SRT.

Example: **−z mgcp,srt**

This option can be used multiple times on the command line.

If the optional *filter* is provided, the stats will only be calculated on those calls that match that filter.

Example: **−z "mgcp,srt,ip.addr==1.2.3.4"** will collect stats only for MGCP packets exchanged by the host at IP address 1.2.3.4 .

−z mtp3,msus[,<filter>]

Show MTP3 MSU statistics.

−z multicast,stat[,<filter>]

Show UDP multicast stream statistics.

−z rpc,programs

Collect call/reply SRT data for all known ONC−RPC programs/versions. Data collected is the number of calls for each protocol/version, MinSRT, MaxSRT and AvgSRT.

−z rpc,srt,*name−or−number*,*version*[,<filter>]

Collect call/reply SRT (Service Response Time) data for program *name*/*version* or *number*/*version*. Data collected is the number of calls for each procedure, MinSRT, MaxSRT and AvgSRT. Program *name* is case−insensitive.

Example: **−z rpc,srt,100003,3** will collect data for NFS v3.

This option can be used multiple times on the command line.

If the optional *filter* is provided, the stats will only be calculated on those calls that match that filter.

Example: **−z rpc,srt,nfs,3,nfs.fh.hash==0x12345678** will collect NFS v3 SRT statistics for a specific file.

−z scsi,srt,*cmdset*[,<filter>]

Collect call/reply SRT (Service Response Time) data for SCSI commandset <cmdset>.

Commandsets are 0:SBC 1:SSC 5:MMC

Data collected is the number of calls for each procedure, MinSRT, MaxSRT and AvgSRT.

Example: **−z scsi,srt,0** will collect data for SCSI BLOCK COMMANDS (SBC).

This option can be used multiple times on the command line.

If the optional *filter* is provided, the stats will only be calculated on those calls that match that filter.

Example: **−z scsi,srt,0,ip.addr==1.2.3.4** will collect SCSI SBC SRT statistics for a specific iscsi/ifcp/fcip host.

−z sip,stat[,*filter*]

This option will activate a counter for SIP messages. You will get the number of occurrences of each SIP Method and of each SIP Status−Code. Additionally you also get the number of resent SIP Messages (only for SIP over UDP).

Example: **−z sip,stat**

This option can be used multiple times on the command line.

If the optional *filter* is provided, the stats will only be calculated on those calls that match that filter.

Example: **−z "sip,stat,ip.addr==1.2.3.4"** will collect stats only for SIP packets exchanged by the host at IP address 1.2.3.4 .

−z smb,srt[,*filter*]

Collect call/reply SRT (Service Response Time) data for SMB. Data collected is the number of calls for each SMB command, MinSRT, MaxSRT and AvgSRT.

Example: **−z smb,srt**

The data will be presented as separate tables for all normal SMB commands, all Transaction2 commands and all NT Transaction commands. Only those commands that are seen in the capture will have their stats displayed. Only the first command in a xAndX command chain will be used in the calculation. So for common SessionSetupAndX + TreeConnectAndX chains, only the SessionSetupAndX call will be used in the statistics. This is a flaw that might be fixed in the future.

This option can be used multiple times on the command line.

If the optional *filter* is provided, the stats will only be calculated on those calls that match that filter.

Example: **−z "smb,srt,ip.addr==1.2.3.4"** will collect stats only for SMB packets exchanged by the host at IP address 1.2.3.4 .

−z voip,calls

This option will show a window that shows VoIP calls found in the capture file. This is the same window shown as when you go to the Statistics Menu and choose VoIP Calls.

Example: **−z voip,calls**

−z wlan,stat[,<filter>]

Show IEEE 802.11 network and station statistics.

−z wsp,stat[,<filter>]

Show WSP packet counters.

INTERFACE

MENU ITEMS

File › Open

File › Open Recent

File › Merge

Merge another capture file to the currently loaded one. The *File:Merge* dialog box allows the merge "Prepended", "Chronologically" or "Appended", relative to the already loaded one.

File › Close

Open or close a capture file. The *File:Open* dialog box allows a filter to be specified; when the capture file is read, the filter is applied to all packets read from the file, and packets not matching the filter are discarded. The *File:Open Recent* is a submenu and will show a list of previously opened files.

File › Save

File › Save As

Save the current capture, or the packets currently displayed from that capture, to a file. Check boxes let you select whether to save all packets, or just those that have passed the current display filter and/or those that are currently marked, and an option menu lets you select (from a list of file formats in which at particular capture, or the packets currently displayed from that capture, can be saved), a file format in which to save it.

File › File Set › List Files

Show a dialog box that lists all files of the file set matching the currently loaded file. A file set is a compound of files resulting from a capture using the "multiple files" / "ringbuffer" mode, recognizable by the filename pattern, e.g.: Filename_00001_20220714101530.pcap.

File › File Set › Next File

File › File Set › Previous File

If the currently loaded file is part of a file set (see above), open the next / previous file in that set.

File › Export

Export captured data into an external format. Note: the data cannot be imported back into Wireshark, so be sure to keep the capture file.

File › Print

Print packet data from the current capture. You can select the range of packets to be printed (which packets are printed), and the output format of each packet (how each packet is printed). The output format will be similar to the displayed values, so a summary line, the packet details view, and/or the hex dump of the packet can be printed.

Printing options can be set with the *Edit:Preferences* menu item, or in the dialog box popped up by this menu item.

File › Quit

Exit the application.

Edit › Copy › Description

Copies the description of the selected field in the protocol tree to the clipboard.

Edit › Copy › Fieldname

Copies the fieldname of the selected field in the protocol tree to the clipboard.

Edit › Copy › Value

Copies the value of the selected field in the protocol tree to the clipboard.

Edit › Copy › As Filter

Create a display filter based on the data currently highlighted in the packet details and copy that filter to the clipboard.

If that data is a field that can be tested in a display filter expression, the display filter will test that field; otherwise, the display filter will be based on the absolute offset within the packet. Therefore it could be unreliable if the packet contains protocols with variable−length headers, such as a source−routed token−ring packet.

Edit › *Find Packet*

Search forward or backward, starting with the currently selected packet (or the most recently selected packet, if no packet is selected). Search criteria can be a display filter expression, a string of hexadecimal digits, or a text string.

When searching for a text string, you can search the packet data, or you can search the text in the Info column in the packet list pane or in the packet details pane.

Hexadecimal digits can be separated by colons, periods, or dashes. Text string searches can be ASCII or Unicode (or both), and may be case insensitive.

Edit › *Find Next*

Edit › *Find Previous*

Search forward / backward for a packet matching the filter from the previous search, starting with the currently selected packet (or the most recently selected packet, if no packet is selected).

Edit › *Mark Packet (toggle)*

Mark (or unmark if currently marked) the selected packet. The field "frame.marked" is set for packets that are marked, so that, for example, a display filters can be used to display only marked packets, and so that the /"Edit:Find Packet" dialog can be used to find the next or previous marked packet.

Edit › *Find Next Mark*

Edit › *Find Previous Mark*

Find next/previous marked packet.

Edit › *Mark All Packets*

Edit › *Unmark All Packets*

Mark / Unmark all packets that are currently displayed.

Edit › *Time Reference* › *Set Time Reference (toggle)*

Set (or unset if currently set) the selected packet as a Time Reference packet. When a packet is set as a Time Reference packet, the timestamps in the packet list pane will be replaced with the string "**REF**". The relative time timestamp in later packets will then be calculated relative to the timestamp of this Time Reference packet and not the first packet in the capture.

Packets that have been selected as Time Reference packets will always be displayed in the packet list pane. Display filters will not affect or hide these packets.

If there is a column displayed for "Cumulative Bytes" this counter will be reset at every Time Reference packet.

Edit › *Time Reference* › *Find Next*

Edit › *Time Reference* › *Find Previous*

Search forward / backward for a time referenced packet.

Edit › *Configuration Profiles*

Manage configuration profiles to be able to use more than one set of preferences and configurations.

Edit › *Preferences*

Set the GUI, capture, printing and protocol options (see /Preferences dialog below).

View › *Main Toolbar*

View › *Filter Toolbar*

View › *Statusbar*

Show or hide the main window controls.

View › *Packet List*

View › *Packet Details*

View › *Packet Bytes*

Show or hide the main window panes.

View › *Time Display Format*

Set the format of the packet timestamp displayed in the packet list window.

View › *Name Resolution* › *Resolve Name*

Try to resolve a name for the currently selected item.

View › *Name Resolution* › *Enable for ... Layer*

Enable or disable translation of addresses to names in the display.

View › *Colorize Packet List*

Enable or disable the coloring rules. Disabling will improve performance.

View › Auto Scroll in Live Capture

Enable or disable the automatic scrolling of the packet list while a live capture is in progress.

View › Zoom In

View › Zoom Out

Zoom into / out of the main window data (by changing the font size).

View › Normal Size

Reset the zoom factor of zoom in / zoom out back to normal font size.

View › Resize All Columns

Resize all columns to best fit the current packet display.

View › Expand / Collapse Subtrees

Expands / Collapses the currently selected item and it's subtrees in the packet details.

View › Expand All

View › Collapse All

Expand / Collapse all branches of the packet details.

View › Colorize Conversation

Select color for a conversation.

View › Reset Coloring 1−10

Reset Color for a conversation.

View › Coloring Rules

Change the foreground and background colors of the packet information in the list of packets, based upon display filters. The list of display filters is applied to each packet sequentially. After the first display filter matches a packet, any additional display filters in the list are ignored. Therefore, if you are filtering on the existence of protocols, you should list the higher−level protocols first, and the lower−level protocols last.

How Colorization Works

Packets are colored according to a list of color filters. Each filter consists of a name, a filter expression and a coloration. A packet is colored according to the first filter that it matches. Color filter expressions use exactly the same syntax as display filter expressions.

When Wireshark starts, the color filters are loaded from:

1. The user’s personal color filters file or, if that does not exist,
2. The global color filters file.

If neither of these exist then the packets will not be colored.

View › Show Packet In New Window

Create a new window containing a packet details view and a hex dump window of the currently selected packet; this window will continue to display that packet’s details and data even if another packet is selected.

View › Reload

Reload a capture file. Same as *File:Close* and *File:Open* the same file again.

Go › Back

Go back in previously visited packets history.

Go › Forward

Go forward in previously visited packets history.

Go › Go To Packet

Go to a particular numbered packet.

Go › Go To Corresponding Packet

If a field in the packet details pane containing a packet number is selected, go to the packet number specified by that field. (This works only if the dissector that put that entry into the packet details put it into the details as a filterable field rather than just as text.) This can be used, for example, to go to the packet for the request corresponding to a reply, or the reply corresponding to a request, if that packet number has been put into the packet details.

Go › Previous Packet

Go › Next Packet

Go › First Packet

Go › Last Packet

Go to the previous / next / first / last packet in the capture.

Go › Previous Packet In Conversation

Go › Next Packet In Conversation

Go to the previous / next packet of the conversation (TCP, UDP or IP)

Capture › Interfaces

Shows a dialog box with all currently known interfaces and displaying the current network traffic amount. Capture sessions can be started from here. Beware: keeping this box open results in high system load!

Capture › Options

Initiate a live packet capture (see /"Capture Options Dialog" below). If no filename is specified, a temporary file will be created to hold the capture. The location of the file can be chosen by setting your TMPDIR environment variable before starting **Wireshark**. Otherwise, the default TMPDIR location is system−dependent, but is likely either */var/tmp* or */tmp*.

Capture › Start

Start a live packet capture with the previously selected options. This won't open the options dialog box, and can be convenient for repeatedly capturing with the same options.

Capture › Stop

Stop a running live capture.

Capture › Restart

While a live capture is running, stop it and restart with the same options again. This can be convenient to remove irrelevant packets, if no valuable packets were captured so far.

Capture › Capture Filters

Edit the saved list of capture filters, allowing filters to be added, changed, or deleted.

Analyze › Display Filters

Edit the saved list of display filters, allowing filters to be added, changed, or deleted.

Analyze › Display Filter Macros

Create shortcuts for complex macros

Analyze › Apply as Filter

Create a display filter based on the data currently highlighted in the packet details and apply the filter.

If that data is a field that can be tested in a display filter expression, the display filter will test that field; otherwise, the display filter will be based on the absolute offset within the packet. Therefore it could be unreliable if the packet contains protocols with variable−length headers, such as a source−routed token−ring packet.

The **Selected** option creates a display filter that tests for a match of the data; the **Not Selected** option creates a display filter that tests for a non−match of the data. The **And Selected**, **Or Selected**, **And Not Selected**, and **Or Not Selected** options add to the end of the display filter in the strip at the top (or bottom) an AND or OR operator followed by the new display filter expression.

Analyze › Prepare as Filter

Create a display filter based on the data currently highlighted in the packet details. The filter strip at the top (or bottom) is updated but it is not yet applied.

Analyze › Enabled Protocols

Allow protocol dissection to be enabled or disabled for a specific protocol. Individual protocols can be enabled or disabled by clicking on them in the list or by highlighting them and pressing the space bar. The entire list can be enabled, disabled, or inverted using the buttons below the list.

When a protocol is disabled, dissection in a particular packet stops when that protocol is reached, and Wireshark moves on to the next packet. Any higher−layer protocols that would otherwise have been processed will not be displayed. For example, disabling TCP will prevent the dissection and display of TCP, HTTP, SMTP, Telnet, and any other protocol exclusively dependent on TCP.

The list of protocols can be saved, so that Wireshark will start up with the protocols in that list disabled.

Analyze › Decode As

If you have a packet selected, present a dialog allowing you to change which dissectors are used to decode this packet. The dialog has one panel each for the link layer, network layer and transport layer protocol/port numbers, and will allow each of these to be changed independently. For example, if the selected packet is a TCP packet to port 12345, using this dialog you can instruct Wireshark to decode all packets to or from that TCP port as HTTP packets.

Analyze › User Specified Decodes

Create a new window showing whether any protocol ID to dissector mappings have been changed by the user. This window also allows the user to reset all decodes to their default values.

Analyze › Follow TCP Stream

If you have a TCP packet selected, display the contents of the data stream for the TCP connection to which that packet belongs, as text, in a separate window, and leave the list of packets in a filtered state, with only those packets that are part of that TCP connection being displayed. You can revert to your old view by pressing ENTER in the display filter text box, thereby invoking your old display filter (or resetting it back to no display filter).

The window in which the data stream is displayed lets you select:

- whether to display the entire conversation, or one or the other side of it;
- whether the data being displayed is to be treated as ASCII or EBCDIC text or as raw hex data;

and lets you print what’s currently being displayed, using the same print options that are used for the *File:Print Packet* menu item, or save it as text to a file.

Analyze › Follow UDP Stream

Analyze › Follow TLS Stream

(Similar to Analyze:Follow TCP Stream)

Analyze › Expert Info

Analyze › Expert Info Composite

(Kind of) a log of anomalies found by Wireshark in a capture file.

Analyze › Conversation Filter

Statistics › Summary

Show summary information about the capture, including elapsed time, packet counts, byte counts, and the like. If a display filter is in effect, summary information will be shown about the capture and about the packets currently being displayed.

Statistics › Protocol Hierarchy

Show the number of packets, and the number of bytes in those packets, for each protocol in the trace. It organizes the protocols in the same hierarchy in which they were found in the trace. Besides counting the packets in which the protocol exists, a count is also made for packets in which the protocol is the last protocol in the stack. These last−protocol counts show you how many packets (and the byte count associated with those packets) **ended** in a particular protocol. In the table, they are listed under "End Packets" and "End Bytes".

Statistics › Conversations

Lists of conversations; selectable by protocol. See Statistics:Conversation List below.

Statistics › End Points

List of End Point Addresses by protocol with packets/bytes/.... counts.

Statistics › Packet Lengths

Grouped counts of packet lengths (0−19 bytes, 20−39 bytes, ...)

Statistics › I/O Graphs

Open a window where up to 5 graphs in different colors can be displayed to indicate number of packets or number of bytes per second for all packets matching the specified filter. By default only one graph will be displayed showing number of packets per second.

The top part of the window contains the graphs and scales for the X and Y axis. If the graph is too long to fit inside the window there is a horizontal scrollbar below the drawing area that can scroll the graphs to the left or the right. The horizontal axis displays the time into the capture and the vertical axis will display the measured quantity at that time.

Below the drawing area and the scrollbar are the controls. On the bottom left there will be five similar sets of controls to control each individual graph such as "Display:<button>" which button will toggle that individual graph on/off. If <button> is ticked, the graph will be displayed. "Color:<color>" which is just a button to show which color will be used to draw that graph. Finally "Filter:<filter−text>" which can be used to specify a display filter for that particular graph.

If filter−text is empty then all packets will be used to calculate the quantity for that graph. If filter−text is specified only those packets that match that display filter will be considered in the calculation of quantity.

To the right of the 5 graph controls there are four menus to control global aspects of the draw area and graphs. The "Unit:" menu is used to control what to measure; "packets/tick", "bytes/tick" or "advanced..."

packets/tick will measure the number of packets matching the (if specified) display filter for the graph in each measurement interval.

bytes/tick will measure the total number of bytes in all packets matching the (if specified) display filter for the graph in each measurement interval.

advanced... see below

"Tick interval:" specifies what measurement intervals to use. The default is 1 second and means that the data will be counted over 1 second intervals.

"Pixels per tick:" specifies how many pixels wide each measurement interval will be in the drawing area. The default is 5 pixels per tick.

"Y−scale:" controls the max value for the y−axis. Default value is "auto" which means that **Wireshark** will try to adjust the maxvalue automatically.

"advanced..." If Unit:advanced... is selected the window will display two more controls for each of the five graphs. One control will be a menu where the type of calculation can be selected from SUM,COUNT,MAX,MIN,AVG and LOAD, and one control, textbox, where the name of a single display filter field can be specified.

The following restrictions apply to type and field combinations:

SUM: available for all types of integers and will calculate the SUM of all occurrences of this field in the measurement interval. Note that some field can occur multiple times in the same packet and then all instances will be summed up. Example: 'tcp.len' which will count the amount of payload data transferred across TCP in each interval.

COUNT: available for all field types. This will COUNT the number of times certain field occurs in each interval. Note that some fields may occur multiple times in each packet and if that is the case then each instance will be counted independently and COUNT will be greater than the number of packets.

MAX: available for all integer and relative time fields. This will calculate the max seen integer/time value seen for the field during the interval. Example: 'smb.time' which will plot the maximum SMB response time.

MIN: available for all integer and relative time fields. This will calculate the min seen integer/time value seen for the field during the interval. Example: 'smb.time' which will plot the minimum SMB response time.

AVG: available for all integer and relative time fields.This will calculate the average seen integer/time value seen for the field during the interval. Example: 'smb.time' which will plot the average SMB response time.

LOAD: available only for relative time fields (response times).

Example of advanced: Display how NFS response time MAX/MIN/AVG changes over time:

Set first graph to:

```
filter:nfs&&rpc.time
Calc:MAX rpc.time
```

Set second graph to

```
filter:nfs&&rpc.time
Calc:AVG rpc.time
```

Set third graph to

```
filter:nfs&&rpc.time
Calc:MIN rpc.time
```

Example of advanced: Display how the average packet size from host a.b.c.d changes over time.

Set first graph to

```
filter:ip.addr==a.b.c.d&&frame.pkt_len
Calc:AVG frame.pkt_len
```

LOAD: The LOAD io−stat type is very different from anything you have ever seen before! While the response times themselves as plotted by MIN,MAX,AVG are indications on the Server load (which affects the Server response time), the LOAD measurement measures the Client LOAD. What this measures is how much workload the client generates, i.e. how fast will the client issue new commands when the previous ones completed. i.e. the level of concurrency the client can maintain. The higher the number, the more and faster is the client issuing new commands. When the LOAD goes down, it may be due to client load making the client slower in issuing new commands (there may be other reasons as well, maybe the client just doesn't have any commands it wants to issue right then).

Load is measured in concurrency/number of overlapping i/o and the value 1000 means there is a constant load of one i/o.

In each tick interval the amount of overlap is measured. See the graph below containing three commands: Below the graph are the LOAD values for each interval that would be calculated.

```
|     |     |     |     |     |     |     |     |
|     |     |     |     |     |     |     |     |
|     |  o=====*  |     |     |     |     |     |
|     |     |     |     |     |     |     |     |
|  o========*     | o============*  |     |     |
|     |     |     |     |     |     |     |     |
--------------------------------------------------> Time
 500   1500   500  750   1000   500    0     0
```

Statistics › *Conversation List*

This option will open a new window that displays a list of all conversations between two endpoints. The list has one row for each unique conversation and displays total number of packets/bytes seen as well as number of packets/bytes in each direction.

By default the list is sorted according to the number of packets but by clicking on the column header; it is possible to re−sort the list in ascending or descending order by any column.

By first selecting a conversation by clicking on it and then using the right mouse button (on those platforms that have a right mouse button) Wireshark will display a popup menu offering several different filter operations to apply to the capture.

These statistics windows can also be invoked from the Wireshark command line using the **−z conv** argument.

Statistics › Service Response Time

- AFP
- CAMEL
- DCE−RPC

Open a window to display Service Response Time statistics for an arbitrary DCE−RPC program interface and display **Procedure**, **Number of Calls**, **Minimum SRT**, **Maximum SRT** and **Average SRT** for all procedures for that program/version. These windows opened will update in semi−real time to reflect changes when doing live captures or when reading new capture files into **Wireshark**.

This dialog will also allow an optional filter string to be used. If an optional filter string is used only such DCE−RPC request/response pairs that match that filter will be used to calculate the statistics. If no filter string is specified all request/response pairs will be used.

- Diameter
- Fibre Channel

Open a window to display Service Response Time statistics for Fibre Channel and display **FC Type**, **Number of Calls**, **Minimum SRT**, **Maximum SRT** and **Average SRT** for all FC types. These windows opened will update in semi−real time to reflect changes when doing live captures or when reading new capture files into **Wireshark**. The Service Response Time is calculated as the time delta between the First packet of the exchange and the Last packet of the exchange.

This dialog will also allow an optional filter string to be used. If an optional filter string is used only such FC first/last exchange pairs that match that filter will be used to calculate the statistics. If no filter string is specified all request/response pairs will be used.

- GTP
- H.225 RAS

Collect requests/response SRT (Service Response Time) data for ITU−T H.225 RAS. Data collected is **number of calls** for each known ITU−T H.225 RAS Message Type, **Minimum SRT**, **Maximum SRT**, **Average SRT**, **Minimum in Packet**, and **Maximum in Packet**. You will also get the number of **Open Requests** (Unresponded Requests), **Discarded Responses** (Responses without matching request) and Duplicate Messages. These windows opened will update in semi−real time to reflect changes when doing live captures or when reading new capture files into **Wireshark**.

You can apply an optional filter string in a dialog box, before starting the calculation. The statistics will only be calculated on those calls matching that filter.

- LDAP
- MEGACO

- MGCP

Collect requests/response SRT (Service Response Time) data for MGCP. Data collected is **number of calls** for each known MGCP Type, **Minimum SRT**, **Maximum SRT**, **Average SRT**, **Minimum in Packet**, and **Maximum in Packet**. These windows opened will update in semi−real time to reflect changes when doing live captures or when reading new capture files into **Wireshark**.

You can apply an optional filter string in a dialog box, before starting the calculation. The statistics will only be calculated on those calls matching that filter.

- NCP
- ONC−RPC

Open a window to display statistics for an arbitrary ONC−RPC program interface and display **Procedure**, **Number of Calls**, **Minimum SRT**, **Maximum SRT** and **Average SRT** for all procedures for that program/version. These windows opened will update in semi−real time to reflect changes when doing live captures or when reading new capture files into **Wireshark**.

This dialog will also allow an optional filter string to be used. If an optional filter string is used only such ONC−RPC request/response pairs that match that filter will be used to calculate the statistics. If no filter string is specified all request/response pairs will be used.

By first selecting a conversation by clicking on it and then using the right mouse button (on those platforms that have a right mouse button) Wireshark will display a popup menu offering several different filter operations to apply to the capture.

- RADIUS
- SCSI
- SMB

Collect call/reply SRT (Service Response Time) data for SMB. Data collected is the number of calls for each SMB command, MinSRT, MaxSRT and AvgSRT.

The data will be presented as separate tables for all normal SMB commands, all Transaction2 commands and all NT Transaction commands. Only those commands that are seen in the capture will have its stats displayed. Only the first command in a xAndX command chain will be used in the calculation. So for common SessionSetupAndX + TreeConnectAndX chains, only the SessionSetupAndX call will be used in the statistics. This is a flaw that might be fixed in the future.

You can apply an optional filter string in a dialog box, before starting the calculation. The stats will only be calculated on those calls matching that filter.

By first selecting a conversation by clicking on it and then using the right mouse button (on those platforms that have a right mouse button) Wireshark will display a popup menu offering several different filter operations to apply to the capture.

- SMB2

Statistics › BOOTP−DHCP

Statistics › Compare

Compare two Capture Files

Statistics › Flow Graph

Flow Graph: General/TCP

Statistics › HTTP

HTTP Load Distribution, Packet Counter & Requests

Statistics › IP Addresses

Count/Rate/Percent by IP Address

Statistics › IP Destinations

Count/Rate/Percent by IP Address/protocol/port

Statistics › IP Protocol Types

Count/Rate/Percent by IP Protocol Types

Statistics › ONC−RPC Programs

This dialog will open a window showing aggregated SRT statistics for all ONC−RPC Programs/versions that exist in the capture file.

Statistics › TCP Stream Graph

Graphs: Round Trip; Throughput; Time−Sequence (Stevens); Time−Sequence (tcptrace)

Statistics › UDP Multicast streams

Multicast Streams Counts/Rates/... by Source/Destination Address/Port pairs

Statistics › WLAN Traffic

WLAN Traffic Statistics

Telephony › ITU−T H.225

Count ITU−T H.225 messages and their reasons. In the first column you get a list of H.225 messages and H.225 message reasons, which occur in the current capture file. The number of occurrences of each message or reason will be displayed in the second column. This window opened will update in semi−real time to reflect changes when doing live captures or when reading new capture files into **Wireshark**.

You can apply an optional filter string in a dialog box, before starting the counter. The statistics will only be calculated on those calls matching that filter.

Telephony › SIP

Activate a counter for SIP messages. You will get the number of occurrences of each SIP Method and of each SIP Status−Code. Additionally you also get the number of resent SIP Messages (only for SIP over UDP).

This window opened will update in semi−real time to reflect changes when doing live captures or when reading new capture files into **Wireshark**.

You can apply an optional filter string in a dialog box, before starting the counter. The statistics will only be calculated on those calls matching that filter.

Tools › *Firewall ACL Rules*

Help › *Contents*

Some help texts.

Help › *Supported Protocols*

List of supported protocols and display filter protocol fields.

Help › *Manual Pages*

Display locally installed HTML versions of these manual pages in a web browser.

Help › *Wireshark Online*

Various links to online resources to be open in a web browser, like https://www.wireshark.org.

Help › *About Wireshark*

See various information about Wireshark (see /About dialog below), like the version, the folders used, the available plugins, ...

WINDOWS

Main Window

The main window contains the usual things like the menu, some toolbars, the main area and a statusbar. The main area is split into three panes, you can resize each pane using a "thumb" at the right end of each divider line.

The main window is much more flexible than before. The layout of the main window can be customized by the *Layout* page in the dialog box popped up by *Edit:Preferences*, the following will describe the layout with the default settings.

Main Toolbar

Some menu items are available for quick access here. There is no way to customize the items in the toolbar, however the toolbar can be hidden by *View:Main Toolbar*.

Filter Toolbar

A display filter can be entered into the filter toolbar. A filter for HTTP, HTTPS, and DNS traffic might look like this:

```
tcp.port in {80 443 53}
```

Selecting the *Filter:* button lets you choose from a list of named filters that you can optionally save. Pressing the Return or Enter keys, or selecting the *Apply* button, will cause the filter to be applied to the current list of packets. Selecting the *Reset* button clears the display filter so that all packets are displayed (again).

There is no way to customize the items in the toolbar, however the toolbar can be hidden by *View:Filter Toolbar*.

Packet List Pane

The top pane contains the list of network packets that you can scroll through and select. By default, the packet number, packet timestamp, source and destination addresses, protocol, and description are displayed for each packet; the *Columns* page in the dialog box popped up by *Edit:Preferences* lets you change this (although, unfortunately, you currently have to save the preferences, and exit and restart Wireshark, for those changes to take effect).

If you click on the heading for a column, the display will be sorted by that column; clicking on the heading again will reverse the sort order for that column.

An effort is made to display information as high up the protocol stack as possible, e.g. IP addresses are displayed for IP packets, but the MAC layer address is displayed for unknown packet types.

The right mouse button can be used to pop up a menu of operations.

The middle mouse button can be used to mark a packet.

Packet Details Pane

The middle pane contains a display of the details of the currently−selected packet. The display shows each field and its value in each protocol header in the stack. The right mouse button can be used to pop up a menu of operations.

Packet Bytes Pane

The lowest pane contains a hex and ASCII dump of the actual packet data. Selecting a field in the packet details highlights the corresponding bytes in this section.

The right mouse button can be used to pop up a menu of operations.

Statusbar

The statusbar is divided into three parts, on the left some context dependent things are shown, like information about the loaded file, in the center the number of packets are displayed, and on the right the current configuration profile.

The statusbar can be hidden by *View:Statusbar*.

Preferences

The *Preferences* dialog lets you control various personal preferences for the behavior of **Wireshark**.

User Interface Preferences

The *User Interface* page is used to modify small aspects of the GUI to your own personal taste:

Selection Bars

The selection bar in the packet list and packet details can have either a "browse" or "select" behavior. If the selection bar has a "browse" behavior, the arrow keys will move an outline of the selection bar, allowing you to browse the rest of the list or details without changing the selection until you press the space bar. If the selection bar has a "select" behavior, the arrow keys will move the selection bar and change the selection to the new item in the packet list or packet details.

Save Window Position

If this item is selected, the position of the main Wireshark window will be saved when Wireshark exits, and used when Wireshark is started again.

Save Window Size

If this item is selected, the size of the main Wireshark window will be saved when Wireshark exits, and used when Wireshark is started again.

Save Window Maximized state

If this item is selected the maximize state of the main Wireshark window will be saved when Wireshark exists, and used when Wireshark is started again.

File Open Dialog Behavior

This item allows the user to select how Wireshark handles the listing of the "File Open" Dialog when opening trace files. "Remember Last Directory" causes Wireshark to automatically position the dialog in the directory of the most recently opened file, even between launches of Wireshark. "Always Open in Directory" allows the user to define a persistent directory that the dialog will always default to.

Directory

Allows the user to specify a persistent File Open directory. Trailing slashes or backslashes will automatically be added.

File Open Preview timeout

This items allows the user to define how much time is spend reading the capture file to present preview data in the File Open dialog.

Open Recent maximum list entries

The File menu supports a recent file list. This items allows the user to specify how many files are kept track of in this list.

Ask for unsaved capture files

When closing a capture file or Wireshark itself if the file isn’t saved yet the user is presented the option to save the file when this item is set.

Wrap during find

This items determines the behavior when reaching the beginning or the end of a capture file. When set the search wraps around and continues, otherwise it stops.

Settings dialogs show a save button

This item determines if the various dialogs sport an explicit Save button or that save is implicit in OK / Apply.

Web browser command

This entry specifies the command line to launch a web browser. It is used to access online content, like the Wiki and user guide. Use '%s' to place the request URL in the command line.

Layout Preferences

The *Layout* page lets you specify the general layout of the main window. You can choose from six different layouts and fill the three panes with the contents you like.

Scrollbars

The vertical scrollbars in the three panes can be set to be either on the left or the right.

Alternating row colors

Hex Display

The highlight method in the hex dump display for the selected protocol item can be set to use either inverse video, or bold characters.

Toolbar style

Filter toolbar placement

Custom window title

Column Preferences

The *Columns* page lets you specify the number, title, and format of each column in the packet list.

The *Column title* entry is used to specify the title of the column displayed at the top of the packet list. The type of data that the column displays can be specified using the *Column format* option menu. The row of buttons on the left perform the following actions:

New

Adds a new column to the list.

Delete

Deletes the currently selected list item.

Up / Down

Moves the selected list item up or down one position.

Font Preferences

The *Font* page lets you select the font to be used for most text.

Color Preferences

The *Colors* page can be used to change the color of the text displayed in the TCP stream window and for marked packets. To change a color, simply select an attribute from the "Set:" menu and use the color selector to get the desired color. The new text colors are displayed as a sample text.

Capture Preferences

The *Capture* page lets you specify various parameters for capturing live packet data; these are used the first time a capture is started.

The *Interface:* combo box lets you specify the interface from which to capture packet data, or the name of a FIFO from which to get the packet data.

The *Data link type:* option menu lets you, for some interfaces, select the data link header you want to see on the packets you capture. For example, in some OSes and with some versions of libpcap, you can choose, on an 802.11 interface, whether the packets should appear as Ethernet packets (with a fake Ethernet header) or as 802.11 packets.

The *Limit each packet to ... bytes* check box lets you set the snapshot length to use when capturing live data; turn on the check box, and then set the number of bytes to use as the snapshot length.

The *Filter:* text entry lets you set a capture filter expression to be used when capturing.

If any of the environment variables SSH_CONNECTION, SSH_CLIENT, REMOTEHOST, DISPLAY, or SESSIONNAME are set, Wireshark will create a default capture filter that excludes traffic from the hosts and ports defined in those variables.

The *Capture packets in promiscuous mode* check box lets you specify whether to put the interface in promiscuous mode when capturing.

The *Update list of packets in real time* check box lets you specify that the display should be updated as packets are seen.

The *Automatic scrolling in live capture* check box lets you specify whether, in an "Update list of packets in real time" capture, the packet list pane should automatically scroll to show the most recently captured packets.

Printing Preferences

The radio buttons at the top of the *Printing* page allow you choose between printing packets with the *File:Print Packet* menu item as text or PostScript, and sending the output directly to a command or saving it to a file. The *Command:* text entry box, on UNIX−compatible systems, is the command to send files to (usually **lpr**), and the *File:* entry box lets you enter the name of the file you wish to save to. Additionally, you can select the *File:* button to browse the file system for a particular save file.

Name Resolution Preferences

The *Enable MAC name resolution*, *Enable network name resolution* and *Enable transport name resolution* check boxes let you specify whether MAC addresses, network addresses, and transport−layer port numbers should be translated to names.

The *Enable concurrent DNS name resolution* allows Wireshark to send out multiple name resolution requests and not wait for the result before continuing dissection. This speeds up dissection with network name resolution but initially may miss resolutions. The number of concurrent requests can be set here as well.

SMI paths

SMI modules

RTP Player Preferences

This page allows you to select the number of channels visible in the RTP player window. It determines the height of the window, more channels are possible and visible by means of a scroll bar.

Protocol Preferences

There are also pages for various protocols that Wireshark dissects, controlling the way Wireshark handles those protocols.

Edit Capture Filter List

Edit Display Filter List

Capture Filter

Display Filter

Read Filter

Search Filter

The *Edit Capture Filter List* dialog lets you create, modify, and delete capture filters, and the *Edit Display Filter List* dialog lets you create, modify, and delete display filters.

The *Capture Filter* dialog lets you do all of the editing operations listed, and also lets you choose or construct a filter to be used when capturing packets.

The *Display Filter* dialog lets you do all of the editing operations listed, and also lets you choose or construct a filter to be used to filter the current capture being viewed.

The *Read Filter* dialog lets you do all of the editing operations listed, and also lets you choose or construct a filter to be used to as a read filter for a capture file you open.

The *Search Filter* dialog lets you do all of the editing operations listed, and also lets you choose or construct a filter expression to be used in a find operation.

In all of those dialogs, the *Filter name* entry specifies a descriptive name for a filter, e.g. **Web and**

DNS traffic. The *Filter string* entry is the text that actually describes the filtering action to take, as described above.The dialog buttons perform the following actions:

New

If there is text in the two entry boxes, creates a new associated list item.

Edit

Modifies the currently selected list item to match what’s in the entry boxes.

Delete

Deletes the currently selected list item.

Add Expression...

For display filter expressions, pops up a dialog box to allow you to construct a filter expression to test a particular field; it offers lists of field names, and, when appropriate, lists from which to select tests to perform on the field and values with which to compare it. In that dialog box, the OK button will cause the filter expression you constructed to be entered into the *Filter string* entry at the current cursor position.

OK

In the *Capture Filter* dialog, closes the dialog box and makes the filter in the *Filter string* entry the filter in the *Capture Preferences* dialog. In the *Display Filter* dialog, closes the dialog box and makes the filter in the *Filter string* entry the current display filter, and applies it to the current capture. In the *Read Filter* dialog, closes the dialog box and makes the filter in the *Filter string* entry the filter in the *Open Capture File* dialog. In the *Search Filter* dialog, closes the dialog box and makes the filter in the *Filter string* entry the filter in the *Find Packet* dialog.

Apply

Makes the filter in the *Filter string* entry the current display filter, and applies it to the current capture.

Save

If the list of filters being edited is the list of capture filters, saves the current filter list to the personal capture filters file, and if the list of filters being edited is the list of display filters, saves the current filter list to the personal display filters file.

Close

Closes the dialog without doing anything with the filter in the *Filter string* entry.

The Color Filters Dialog

This dialog displays a list of color filters and allows it to be modified.

THE FILTER LIST

Single rows may be selected by clicking. Multiple rows may be selected by using the ctrl and shift keys in combination with the mouse button.

NEW

Adds a new filter at the bottom of the list and opens the Edit Color Filter dialog box. You will have to alter the filter expression at least before the filter will be accepted. The format of color filter expressions is identical to that of display filters. The new filter is selected, so it may immediately be moved up and down, deleted or edited. To avoid confusion all filters are unselected before the new filter is created.

EDIT

Opens the Edit Color Filter dialog box for the selected filter. (If this button is disabled you may have more than one filter selected, making it ambiguous which is to be edited.)

ENABLE

Enables the selected color filter(s).

DISABLE

Disables the selected color filter(s).

DELETE

Deletes the selected color filter(s).

EXPORT

Allows you to choose a file in which to save the current list of color filters. You may also choose to save only the selected filters. A button is provided to save the filters in the global color filters file (you must have sufficient permissions to write this file, of course).

IMPORT

Allows you to choose a file containing color filters which are then added to the bottom of the current list. All the added filters are selected, so they may be moved to the correct position in the list as a group. To avoid confusion, all filters are unselected before the new filters are imported. A button is provided to load the filters from the global color filters file.

CLEAR

Deletes your personal color filters file, reloads the global color filters file, if any, and closes the dialog.

UP

Moves the selected filter(s) up the list, making it more likely that they will be used to color packets.

DOWN

Moves the selected filter(s) down the list, making it less likely that they will be used to color packets.

OK

Closes the dialog and uses the color filters as they stand.

APPLY

Colors the packets according to the current list of color filters, but does not close the dialog.

SAVE

Saves the current list of color filters in your personal color filters file. Unless you do this they will not be used the next time you start Wireshark.

CLOSE

Closes the dialog without changing the coloration of the packets. Note that changes you have made to the current list of color filters are not undone.

Capture Options Dialog

The *Capture Options Dialog* lets you specify various parameters for capturing live packet data.

The *Interface:* field lets you specify the interface from which to capture packet data or a command from which to get the packet data via a pipe.

The *Link layer header type:* field lets you specify the interfaces link layer header type. This field is usually disabled, as most interface have only one header type.

The *Capture packets in promiscuous mode* check box lets you specify whether the interface should be put into promiscuous mode when capturing.

The *Limit each packet to ... bytes* check box and field lets you specify a maximum number of bytes per packet to capture and save; if the check box is not checked, the limit will be 262144 bytes.

The *Capture Filter:* entry lets you specify the capture filter using a tcpdump−style filter string as described above.

The *File:* entry lets you specify the file into which captured packets should be saved, as in the *Printer Options* dialog above. If not specified, the captured packets will be saved in a temporary file; you can save those packets to a file with the *File:Save As* menu item.

The *Use multiple files* check box lets you specify that the capture should be done in "multiple files" mode. This option is disabled, if the *Update list of packets in real time* option is checked.

The *Next file every ... megabyte(s)* check box and fields lets you specify that a switch to a next file should be done if the specified filesize is reached. You can also select the appropriate unit, but beware that the filesize has a maximum of 2 GiB. The check box is forced to be checked, as "multiple files" mode requires a file size to be specified.

The *Next file every ... minute(s)* check box and fields lets you specify that the switch to a next file should be done after the specified time has elapsed, even if the specified capture size is not reached.

The *Ring buffer with ... files* field lets you specify the number of files of a ring buffer. This feature will capture into the first file again, after the specified number of files have been used.

The *Stop capture after ... files* field lets you specify the number of capture files used, until the capture is stopped.

The *Stop capture after ... packet(s)* check box and field let you specify that Wireshark should stop capturing after having captured some number of packets; if the check box is not checked, Wireshark will not stop capturing at some fixed number of captured packets.

The *Stop capture after ... megabyte(s)* check box and field lets you specify that Wireshark should stop capturing after the file to which captured packets are being saved grows as large as or larger than some specified number of megabytes. If the check box is not checked, Wireshark will not stop capturing at some capture file size (although the operating system on which Wireshark is running, or the available disk space, may still limit the maximum size of a capture file). This option is disabled, if "multiple files" mode is used,

The *Stop capture after ... second(s)* check box and field let you specify that Wireshark should stop capturing after it has been capturing for some number of seconds; if the check box is not checked, Wireshark will not stop capturing after some fixed time has elapsed.

The *Update list of packets in real time* check box lets you specify whether the display should be updated as packets are captured and, if you specify that, the *Automatic scrolling in live capture* check box lets you specify the packet list pane should automatically scroll to show the most recently captured packets as new packets arrive.

The *Enable MAC name resolution*, *Enable network name resolution* and *Enable transport name resolution* check boxes let you specify whether MAC addresses, network addresses, and transport−layer port numbers should be translated to names.

About

The *About* dialog lets you view various information about Wireshark.

About › Wireshark

The *Wireshark* page lets you view general information about Wireshark, like the installed version, licensing information and such.

About › Authors

The *Authors* page shows the author and all contributors.

About › Folders

The *Folders* page lets you view the directory names where Wireshark is searching it’s various configuration and other files.

About › Plugins

The *Plugins* page lets you view the dissector plugin modules available on your system.

The *Plugins List* shows the name and version of each dissector plugin module found on your system.

On Unix−compatible systems, the plugins are looked for in the following directories: the *lib/wireshark/plugins/$VERSION* directory under the main installation directory (for example, */usr/local/lib/wireshark/plugins/$VERSION*), and then *$HOME/.wireshark/plugins*.

On Windows systems, the plugins are looked for in the following directories: *plugins\$VERSION* directory under the main installation directory (for example, *C:\Program*

Files\Wireshark\plugins\$VERSION), and then *%APPDATA%\Wireshark\plugins\$VERSION* (or, if %APPDATA% isn't defined, *%USERPROFILE%\Application Data\Wireshark\plugins\$VERSION*).

$VERSION is the version number of the plugin interface, which is typically the version number of Wireshark. Note that a dissector plugin module may support more than one protocol; there is not necessarily a one−to−one correspondence between dissector plugin modules and protocols. Protocols supported by a dissector plugin module are enabled and disabled using the *Edit:Protocols* dialog box, just as protocols built into Wireshark are.

CAPTURE FILTER SYNTAX

See the manual page of pcap−filter(7) or, if that doesn't exist, tcpdump(8), or, if that doesn't exist, https://gitlab.com/wireshark/wireshark/−/wikis/CaptureFilters.

DISPLAY FILTER SYNTAX

For a complete table of protocol and protocol fields that are filterable in **Wireshark** see the wireshark−filter(4) manual page.

FILES

These files contains various **Wireshark** configuration settings.

Preferences

The *preferences* files contain global (system−wide) and personal preference settings. If the system−wide preference file exists, it is read first, overriding the default settings. If the personal preferences file exists, it is read next, overriding any previous values. Note: If the command line flag **−o** is used (possibly more than once), it will in turn override values from the preferences files.

The preferences settings are in the form *prefname:value*, one per line, where *prefname* is the name of the preference and *value* is the value to which it should be set; white space is allowed between **:** and *value*. A preference setting can be continued on subsequent lines by indenting the continuation lines with white space. A **#** character starts a comment that runs to the end of the line:

```
# Vertical scrollbars should be on right side?
# TRUE or FALSE (case−insensitive).
gui.scrollbar_on_right: TRUE
```

The global preferences file is looked for in the *wireshark* directory under the *share* subdirectory of the main installation directory (for example, */usr/local/share/wireshark/preferences*) on UNIX−compatible systems, and in the main installation directory (for example, *C:\Program Files\Wireshark\preferences*) on Windows systems.

The personal preferences file is looked for in *$XDG_CONFIG_HOME/wireshark/preferences* (or, if *$XDG_CONFIG_HOME/wireshark* does not exist while *$HOME/.wireshark* is present, *$HOME/.wireshark/preferences*) on UNIX−compatible systems and *%APPDATA%\Wireshark\preferences* (or, if %APPDATA% isn't defined, *%USERPROFILE%\Application Data\Wireshark\preferences*) on Windows systems.

Note: Whenever the preferences are saved by using the *Save* button in the *Edit:Preferences* dialog box, your personal preferences file will be overwritten with the new settings, destroying any comments and unknown/obsolete settings that were in the file.

Recent

The *recent* file contains personal settings (mostly GUI related) such as the current **Wireshark** window size. The file is saved at program exit and read in at program start automatically. Note: The command line flag **−o** may be used to override settings from this file.

The settings in this file have the same format as in the *preferences* files, and the same directory as for the personal preferences file is used.

Note: Whenever Wireshark is closed, your recent file will be overwritten with the new settings, destroying any comments and unknown/obsolete settings that were in the file.

Disabled (Enabled) Protocols

The *disabled_protos* files contain system−wide and personal lists of protocols that have been disabled, so that their dissectors are never called. The files contain protocol names, one per line, where the protocol name is the same name that would be used in a display filter for the protocol:

```
http
tcp     # a comment
```

If a protocol is listed in the global *disabled_protos* file, it is not displayed in the *Analyze:Enabled Protocols* dialog box, and so cannot be enabled by the user.

The global *disabled_protos* file uses the same directory as the global preferences file.

The personal *disabled_protos* file uses the same directory as the personal preferences file.

Note: Whenever the disabled protocols list is saved by using the *Save* button in the *Analyze:Enabled Protocols* dialog box, your personal disabled protocols file will be overwritten with the new settings, destroying any comments that were in the file.

Name Resolution (hosts)

If the personal *hosts* file exists, it is used to resolve IPv4 and IPv6 addresses before any other attempts are made to resolve them. The file has the standard *hosts* file syntax; each line contains one IP address and name, separated by whitespace. The same directory as for the personal preferences file is used.

Capture filter name resolution is handled by libpcap on UNIX−compatible systems and WinPcap on Windows. As such the Wireshark personal *hosts* file will not be consulted for capture filter name resolution.

Name Resolution (subnets)

If an IPv4 address cannot be translated via name resolution (no exact match is found) then a partial match is attempted via the *subnets* file. Both the global *subnets* file and personal *subnets* files are used if they exist.

Each line of this file consists of an IPv4 address, a subnet mask length separated only by a / and a name separated by whitespace. While the address must be a full IPv4 address, any values beyond the mask length are subsequently ignored.

An example is:

# Comments must be prepended by the # sign! 192.168.0.0/24 ws_test_network

A partially matched name will be printed as "subnet−name.remaining−address". For example, "192.168.0.1" under the subnet above would be printed as "ws_test_network.1"; if the mask length above had been 16 rather than 24, the printed address would be "ws_test_network.0.1".

Name Resolution (ethers)

The *ethers* files are consulted to correlate 6−byte hardware addresses to names. First the personal *ethers* file is tried and if an address is not found there the global *ethers* file is tried next.

Each line contains one hardware address and name, separated by whitespace. The digits of the hardware address are separated by colons (:), dashes (−) or periods (.). The same separator character must be used consistently in an address. The following three lines are valid lines of an *ethers* file:

```
ff:ff:ff:ff:ff:ff          Broadcast
c0−00−ff−ff−ff−ff          TR_broadcast
00.00.00.00.00.00          Zero_broadcast
```

The global *ethers* file is looked for in the */etc* directory on UNIX−compatible systems, and in the main installation directory (for example, *C:\Program Files\Wireshark*) on Windows systems.

The personal *ethers* file is looked for in the same directory as the personal preferences file.

Capture filter name resolution is handled by libpcap on UNIX−compatible systems and WinPcap on Windows. As such the Wireshark personal *ethers* file will not be consulted for capture filter name resolution.

Name Resolution (manuf)

The *manuf* file is used to match the 3−byte vendor portion of a 6−byte hardware address with the manufacturer's name; it can also contain well−known MAC addresses and address ranges specified with a netmask. The format of the file is the same as the *ethers* files, except that entries such as:

```
00:00:0C      Cisco
```

can be provided, with the 3−byte OUI and the name for a vendor, and entries such as:

```
00−00−0C−07−AC/40     All−HSRP−routers
```

can be specified, with a MAC address and a mask indicating how many bits of the address must match. The above entry, for example, has 40 significant bits, or 5 bytes, and would match addresses from 00−00−0C−07−AC−00 through 00−00−0C−07−AC−FF. The mask need not be a multiple of 8.

The *manuf* file is looked for in the same directory as the global preferences file.

Name Resolution (services)

The *services* file is used to translate port numbers into names. Both the global *services* file and personal *services* files are used if they exist.

The file has the standard *services* file syntax; each line contains one (service) name and one transport identifier separated by white space. The transport identifier includes one port number and one transport protocol name (typically tcp, udp, or sctp) separated by a /.

An example is:

mydns   5045/udp   # My own Domain Name Server mydns   5045/tcp   # My own Domain Name Server

Name Resolution (ipxnets)

The *ipxnets* files are used to correlate 4−byte IPX network numbers to names. First the global *ipxnets* file is tried and if that address is not found there the personal one is tried next.

The format is the same as the *ethers* file, except that each address is four bytes instead of six. Additionally, the address can be represented as a single hexadecimal number, as is more common in the IPX world, rather than four hex octets. For example, these four lines are valid lines of an *ipxnets* file:

```
C0.A8.2C.00              HR
c0−a8−1c−00              CEO
00:00:BE:EF              IT_Server1
110f                     FileServer3
```

The global *ipxnets* file is looked for in the */etc* directory on UNIX−compatible systems, and in the main installation directory (for example, *C:\Program Files\Wireshark*) on Windows systems.

The personal *ipxnets* file is looked for in the same directory as the personal preferences file.

Capture Filters

The *cfilters* files contain system−wide and personal capture filters. Each line contains one filter, starting with the string displayed in the dialog box in quotation marks, followed by the filter string itself:

```
"HTTP" port 80
"DCERPC" port 135
```

The global *cfilters* file uses the same directory as the global preferences file.

The personal *cfilters* file uses the same directory as the personal preferences file. It is written through the Capture:Capture Filters dialog.

If the global *cfilters* file exists, it is used only if the personal *cfilters* file does not exist; global and personal capture filters are not merged.

Display Filters

The *dfilters* files contain system−wide and personal display filters. Each line contains one filter, starting with the string displayed in the dialog box in quotation marks, followed by the filter string itself:

```
"HTTP" http
"DCERPC" dcerpc
```

The global *dfilters* file uses the same directory as the global preferences file.

The personal *dfilters* file uses the same directory as the personal preferences file. It is written through the Analyze:Display Filters dialog.

If the global *dfilters* file exists, it is used only if the personal *dfilters* file does not exist; global and personal display filters are not merged.

Color Filters (Coloring Rules)

The *colorfilters* files contain system−wide and personal color filters. Each line contains one filter, starting with the string displayed in the dialog box, followed by the corresponding display filter. Then the background and foreground colors are appended:

```
# a comment
@tcp@tcp@[59345,58980,65534][0,0,0]
@udp@udp@[28834,57427,65533][0,0,0]
```

The global *colorfilters* file uses the same directory as the global preferences file.

The personal *colorfilters* file uses the same directory as the personal preferences file. It is written through the View:Coloring Rules dialog.

If the global *colorfilters* file exists, it is used only if the personal *colorfilters* file does not exist; global and personal color filters are not merged.

Plugins

See above in the description of the About:Plugins page.

ENVIRONMENT VARIABLES

WIRESHARK_CONFIG_DIR

This environment variable overrides the location of personal configuration files. It defaults to *$XDG_CONFIG_HOME/wireshark* (or *$HOME/.wireshark* if the former is missing while the latter exists). On Windows, *%APPDATA%\Wireshark* is used instead. Available since Wireshark 3.0.

WIRESHARK_DEBUG_WMEM_OVERRIDE

Setting this environment variable forces the wmem framework to use the specified allocator backend for **all** allocations, regardless of which backend is normally specified by the code. This is mainly useful to developers when testing or debugging. See *README.wmem* in the source distribution for details.

WIRESHARK_RUN_FROM_BUILD_DIRECTORY

This environment variable causes the plugins and other data files to be loaded from the build directory (where the program was compiled) rather than from the standard locations. It has no effect when the program in question is running with root (or setuid) permissions on *NIX.

WIRESHARK_DATA_DIR

This environment variable causes the various data files to be loaded from a directory other than the standard locations. It has no effect when the program in question is running with root (or setuid) permissions on *NIX.

ERF_RECORDS_TO_CHECK

This environment variable controls the number of ERF records checked when deciding if a file really is in the ERF format. Setting this environment variable a number higher than the default (20) would make false positives less likely.

IPFIX_RECORDS_TO_CHECK

This environment variable controls the number of IPFIX records checked when deciding if a file really is in the IPFIX format. Setting this environment variable a number higher than the default (20) would make false positives less likely.

WIRESHARK_ABORT_ON_DISSECTOR_BUG

If this environment variable is set, **Wireshark** will call abort(3) when a dissector bug is encountered. abort(3) will cause the program to exit abnormally; if you are running **Wireshark** in a debugger, it should halt in the debugger and allow inspection of the process, and, if you are not running it in a debugger, it will, on some OSes, assuming your environment is configured correctly, generate a core dump file. This can be useful to developers attempting to troubleshoot a problem with a protocol dissector.

WIRESHARK_ABORT_ON_TOO_MANY_ITEMS

If this environment variable is set, **Wireshark** will call abort(3) if a dissector tries to add too many items to a tree (generally this is an indication of the dissector not breaking out of a loop soon enough). abort(3) will cause the program to exit abnormally; if you are running **Wireshark** in a debugger, it should halt in the debugger and allow inspection of the process, and, if you are not running it in a debugger, it will, on some OSes, assuming your environment is configured correctly, generate a core dump file. This can be useful to developers attempting to troubleshoot a problem with a protocol dissector.

WIRESHARK_QUIT_AFTER_CAPTURE

Cause **Wireshark** to exit after the end of the capture session. This doesn’t automatically start a capture; you must still use **−k** to do that. You must also specify an autostop condition, e.g. **−c** or **−a duration:...**. This means that you will not be able to see the results of the capture after it stops; it’s primarily useful for testing.

WIRESHARK_LOG_LEVEL

This environment variable controls the verbosity of diagnostic messages to the console. From less verbose to most verbose levels can be `critical`, `warning`, `message`, `info`, `debug` or `noisy`. Levels above the current level are also active. Levels `critical` and `error` are always active.

WIRESHARK_LOG_FATAL

Sets the fatal log level. Fatal log levels cause the program to abort. This level can be set to `Error`, `critical` or `warning`. `Error` is always fatal and is the default.

WIRESHARK_LOG_DOMAINS

This environment variable selects which log domains are active. The filter is given as a case−insensitive comma separated list. If set only the included domains will be enabled. The default domain is always considered to be enabled. Domain filter lists can be preceded by '!' to invert the sense of the match.

WIRESHARK_LOG_DEBUG

List of domains with `debug` log level. This sets the level of the provided log domains and takes precedence over the active domains filter. If preceded by '!' this disables the `debug` level instead.

WIRESHARK_LOG_NOISY

Same as above but for `noisy` log level instead.

AUTHORS

Wireshark would not be the powerful, featureful application it is without the generous contributions of hundreds of developers.

A complete list of authors can be found in the AUTHORS file in Wireshark's source code repository and at https://www.wireshark.org/about.html#authors.

SEE ALSO

wireshark−filter(4), tshark(1), editcap(1), pcap(3), dumpcap(1), mergecap(1), text2pcap(1), pcap−filter(7) or tcpdump(8)

NOTES

This is the manual page for **Wireshark** 3.6.2. The latest version of **Wireshark** can be found at https://www.wireshark.org.

HTML versions of the Wireshark project man pages are available at https://www.wireshark.org/docs/man−pages.