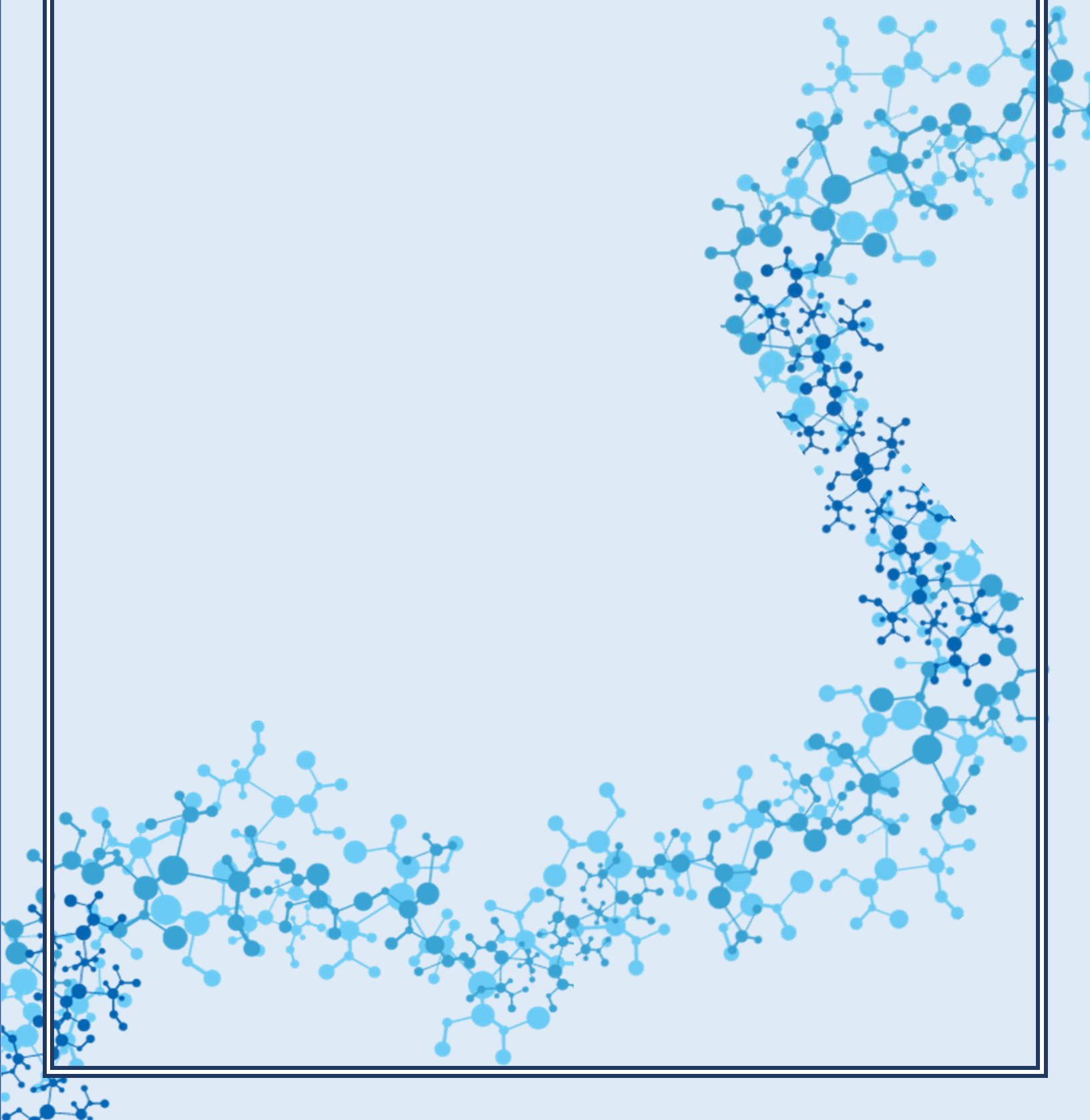
# PCAP TRACE PROCESSING TOOL: MANUAL



# OVERVIEW:

This documentation provides a detailed run through into what is regarded as the PCAP Trace Processing Tool, abbreviated as PTPT. This tool is designed to provide insight on a network's performance, and correspondingly user behavior.

The PTPT's capabilities range from processing network traces at the packet and flow level, calculating important data points, and visualizing statistics in several ways. The tool is designed to provide an efficient way to view select, sought after data points without the burden of loading unnecessary extras. Calculating specific, aggregate values without complication is present, as values can be obtained in a straightforward manner with simple commands. And the further utilization of visualization features permits an additional way to view values instead of solely relying on the numbers.

As is the software, this manual will be divided into three main sections: Process, Analysis, and Visualization. Each section will provide an in-depth description of the features each component possesses, and how to make use of these features. Examples will be provided for further clarification. Before this however, a Set-Up section is included which parallels the included README documentation. This will provide all the information on necessary installations and environment settings to permit the successful functioning of this tool. There also is a General section to briefly review start-up. At the end of this manual, there exists an Additional Information, Background, and Appendix section for further information on select topics.

# SET-UP:

## Installations (software):

- **Wireshark** (https://www.wireshark.org/#download)
- **Tshark** (https://www.wireshark.org/#download; **comes installed with Wireshark*)
- **AWK** (http://gnuwin32.sourceforge.net/packages/gawk.htm)

## Installations (libraries/packages):

- **NumPY** ('pip install numpy')
- **Matplotlib** ('pip install matplotlib')
- **Plotly** ('pip install plotly')
- **Kaleido** ('pip install kaleido')

## Environment Settings:

- The path to Wireshark should be added to the PATH environment variable under system settings. Without this, Tshark commands to process network traces cannot be successfully executed. For Windows, this can be found under 'System Properties -> Environment Variables -> System Variables'. This is the recommended approach, however an alternative would be to add the full path to Wireshark in the TVWS_process script instead.

- Wireshark's settings should ideally be set to default. Some settings, such as name resolution under 'Edit -> Preferences -> Name Resolution' are overridden and will not cause complications when running the software, however other changes may cause unforeseen issues.
- The Python libraries need to be updated to the latest version.

# GENERAL:

Upon start-up, the user will be on the Main Menu. From here, the user can enter the Settings menu via the command **'settings'** as well as process traces and access various tools. Processing traces will be described in the Process section, whilst using tools will be explained in both the Analysis and Visualization sections. Users can also exit the application with **'exit'**.

In the Settings menu, the user may set an input directory, output directory, specify date filters, display filters, and output fields.

The input directory is the directory where .pcap files should be located when they are to be processed, as well as where .csv files should be when performing calculations and data visualization. This can be set with the formatted command: **'indir <directory path>'**.

The output directory is the directory where processed files should go to. This directory is not considered when performing calculations or data visualization (except if the user wishes to save visualization outputs). This can be set with the formatted command: **'outdir <directory path>'**. The input directory and output directory may be the same.

With the filter command, a date range may be specified, along with display filters to narrow down packets in a trace and fields that are specifically desired.

When a date range is specified, only .pcap files within that date range will be processed, and only .csv files within that date range will be considered for data point calculations and visualization. This can be set with the formatted command: **'filter date <date>-<date>'**. The date is the year followed by the month followed by the day followed by the hour. An example of an acceptable, valid input would be **'filter date 2020010100-2020013123'**. (*Note: The current code checks the file's name from the $12^{th}$ index to the $22^{nd}$. If the date is present on another range, this must be altered in the relevant scripts, which can be simply done)

When a display filter is specified, .pcap files will be processed after they are filtered based on the specified display filter. Currently, the only two Wireshark display filters that can be utilized are 'tcp' and 'udp', to process the packets based on transport protocol. This can be set with the formatted command: **'filter display <transport protocol>'**.

When one or more fields are specified, .pcap files are processed with just these fields in mind. There are many supported fields for packet data extraction, of which the full list can be seen in the Appendix section at the end of this manual. Fields can be set with the formatted command: **'filter field <fields (separated by space)>'**. An example input would be **'filter field ip.src ip.dst ip.ttl'**.

## PROCESS:

At the Main Menu, .pcap files can be processed at the packet or flow level. This can be done with the formatted command: **'process"** or **"process_<transport protocol>flows'**.

When processing, it is required that an input and output directory has been set via the Settings menu.

It is also essential that .pcap files processed at the packet and flow level are separated; if there are 10 .pcap files in directory A, .csv files from **'process'** should be sent to directory B whilst .csv files from **'process_<transport protocol>flows'** should be sent to directory C. In essence, these files should never be merged as they will not be differentiated by the software.

## ANALYSIS:

After traces are processed, they can be analyzed extensively. At the Main Menu, entering **'tools'** will direct the user to the Tools Menu, where they can perform calculations to obtain many data points associated with the processed traces.

With the calc command, users can specify a function and the field to perform the calculation on. This can be done with the formatted command: **'calc <function type> <field> <code/service/site (if required)>'**.

There are three functions: 'count', which obtains the occurrences of a field, 'total' which takes the sum of a field, and 'avg', which takes the average of a field.

For fields, any fields inputted in the Settings Menu can be utilized here, as well as many more that pertain to flows. Again, the full list can be found in the Appendix section.

Commands with certain fields will require an additional input. For commands associated with HTTP response codes, the response code number must be inputted. For commands associated with services accessed, the service must be inputted. And for commands associated with popular sites visited, the domain must be inputted. Not all inputs are supported, although a sizeable list of valid inputs regarding these fields are available and listed in the Appendix. Example inputs are as follows:

- **'calc total frame.len'** – Obtains the sum of data in all packets
- **'calc count http.response.code 200'** – Obtains the number of times the 200 response code occurs
- **'calc avg servicesbytes_tcpflows HTTPS'** – Obtains the average bytes associated with a HTTPS flow
- **'calc count site_tcpflows Google'** – Obtains the number of times Google is visited

Of course, certain commands are not possible. For instance, it would not make sense to permit the calculation of the sum of source ports, so this is not allowed.

Calculated data points are stored via a dictionary. Thus, should a calculation be repeated, the tool will save time by retrieving the value rather than re-calculating. However, should the user wish to

re-calculate a data point, they can preface their commands with 'rep'. For example, **'rep calc total frame.len'** will re-calculate the sum of data in all packets. (*Note: Some data points, particularly ones that require an additional input such has HTTP response codes, need to use 'rep' for every calculation after the first)

(*Note 2: 1-packet flows are disregarded when calculating successful/failed flows, on the premise of the presence of a FIN packet. Also, sites accessed are calculated using a reverse DNS lookup, which does not always fetch the domain name)

(*Note 3: It is necessary that the directory of the related, processed files be used when performing calculations. This means for calculations pertaining to base fields, having the input directory set as the path to the processed traces at the packet-level, whereas with calculations pertaining to flows, having the input directory set as the path to the processed traces at the flow-level)

# VISUALIZATION:

There are three ways processed and calculated data can be visualized: plotted against a cumulative distribution function, via a bar graph, or on a table. Similarly with the Analysis component, data visualization is performed within the Tools Menu. The command should be prefixed with 'graph' following the desired visualization.

For plotting against a CDF, this type of graph will show the distribution of each of the values obtained from a specific field. Within the y-axis, the probability of a certain value will be graphed showing how probable that value is going to be. This probability will range from 0 to 1 with 1 being the most probable and 0 being the least probable. Within the x-axis, this will show the range of values obtained from the selected field to be graphed. Within this CDF, the probability will be the sum of each of the previous probabilities, this will show how probable it is to obtain at or less than each specific x-axis value that was obtained through the processing.

The format of a valid input is as follows: **'graph <visualization type> <pre-defined visual (optional)>'**. Pre-defined visuals refer to the set of pre-defined outputs for particular fields. Several are available for bar graphs and tables. They are:

**Bar Graphs**:

- **'figure7'**: Outputs percentage of flow count and byte total of popular services used
- **'figure8'**: Outputs percentages of flow count of a medium-sized list of popular websites used
- **'figure9'**: Outputs percentages of flow count and byte total of select popular websites accessed

**Tables**:

- **'table1'**: Outputs general statistics at the packet-level
- **'table2'**: Outputs general statistics at the flow-level
- **'table3'**: Outputs statistics at the flow-level, emphasis on uplink and downlink characteristics
- **'table4'**: Outputs HTTP response code counts (200, 400, 408)
- **'table6'**: Outputs general statistics at the flow-level

Further, CDF graphs have pre-defined outputs for select fields, including 'frame.len', 'frame.time_delta', 'tcp.len', 'tcp.time_delta', 'ip.ttl', 'tcp.analysis.ack_rtt', 'tcp.analysis.bytes_in_flight', and 'tcp.window_size'. These outputs have been hardcoded for better appearance.

Examples of formatted inputs would be:

- **'graph CDF tcp.analysis.bytes_in_flight'**
- **'graph bar'**
- **'graph bar figure7'**
- **'graph table'**
- **'graph table table2'**

(*Note: Just as with analysis, it is necessary that the directory of the related files be used when performing visualization. This means for outputs pertaining to base fields, having the input directory set as the path to the processed traces at the packet-level, whereas with outputs pertaining to flows, having the input directory set as the path to the processed traces at the flow-level)

(*Note 2: To have outputs saved as .png files, an output directory must be set via the Settings menu)

## ADDITIONAL INFORMATION:

At any input prompt, the user may enter **'info'** for the required, valid format and corresponding description.

If the current list of fields is not expansive enough and should the user wish to add additional fields that can be used with Tshark, this can easily be done: in the TVWS_main_filters file, simply add the field to the relevant lists (i.e. if it is a base field, add it to the 'fieldFilters' list, and if a count calculation can be performed on it, and it to the 'countFIlters' list as well, and so on).

## BACKGROUND:

Aimed at understanding Television White Space Networks and how well the technology performed, this tool was born from CSI 499 at the University at Albany. CSI 499 is the Capstone Project, the course taken by undergraduate seniors which has students demonstrate their knowledge and skills obtained through the computer science undergraduate program, in a team setting.

The project was sponsored by Professor Mariya Zheleva and the UbiNET Lab at SUNY Albany. Instruction and insight were provided by Professor Mariya Zheleva and PhD student Vaasu Taneja. This tool was developed by Munthir Chater with contributions by Jace O'Connor and Keenen Topple. This manual was written by Munthir Chater with contributions by Jace O'Connor.

After development, research was conducted on approximately 1800 network traces taken via a TVWS Network in upstate New York between the months of November 2017 and April 2018. This research was presented in the Spring 2022 semester, at the conclusion of the course.

# APPENDIX:

The full list of base fields that can be specified to be extracted from packets in a trace when using the 'process' command and in analysis and data visualization are:

**Base Fields:**

"frame.time_epoch", "frame.time_delta", "frame.len", "ip.src", "ip.dst", "eth.src", "eth.dst", "tcp.len", "ip.proto", "ip.ttl", "tcp.srcport", "tcp.dstport", "udp.srcport", "udp.dstport", "tcp.flags", "tcp.flags.ack", "tcp.flags.syn", "tcp.flags.fin", "tcp.analysis.ack_rtt", "tcp.analysis.retransmission", "tcp.time_delta", "udp.time_delta", "tcp.analysis.bytes_in_flight", "tcp.window_size", "http.request", "http.request.uri", "http.request.full_uri", "http.host", "http.request.uri.path", "http.request.uri.query", "http.response.code", "icmp.type"

The full list of additional fields associated with TCP and UDP flows that can be used in analysis and data visualization are:

**TCP Flows Fields:**

"flw_tcpflows", "successfulTCPFlows", "failedTCPFlows", "pkt_tcpflows", "uppkt_tcpflows", "dwpkt_tcpflows", "flwsiz_tcpflows", "upflwsiz_tcpflows", "dwflwsiz_tcpflows", "flwsiz_successfulTCPFlows", "flwsiz_failedTCPFlows", "pktsiz_tcpflows", "uppktsiz_tcpflows", "dwpktsiz_tcpflows", "pkts_successfulTCPFlows", "pkts_failedTCPFlows", "services_tcpflows", "servicespackets_tcpflows", "servicesbytes_tcpflows", "servicesupbytes_tcpflows", "servicesdwbytes_tcpflows", "site_tcpflows", "siteSuccess_tcpflows", "siteFail_tcpflows", "sitePackets_tcpflows", "siteSuccessPackets_tcpflows", "siteFailPackets_tcpflows", "siteBytes_tcpflows", "siteSuccessBytes_tcpflows", "siteFailBytes_tcpflows"

**UDP Flow Fields:**

"flw_udpflows", "pkt_udpflows", "uppkt_udpflows", "dwpkt_udpflows", "flwsiz_udpflows", "upflwsiz_udpflows", "dwflwsiz_udpflows", "pktsiz_udpflows", "uppktsiz_udpflows", "dwpktsiz_udpflows", "services_udpflows", "servicespackets_udpflows", "servicesbytes_udpflows", "servicesupbytes_udpflows", "servicesdwbytes_udpflows", "site_udpflows", "sitePackets_udpflows", "siteBytes_udpflows"

The full list of supported HTTP response codes, services, and websites are:

**HTTP Response Codes:** "100", "200", "204", "206", "300", "301", "302", "303", "304", "400", "401", "403", "404", "408", "500", "501", "503"

**Services:** "HTTP", "SSL", "HTTPS", "FTP", "SFTP", "SSH", "FTPs", "MySQL", "Telnet", "SMPTP", "DNS", "BitTorrent", "Napster", "Gnutella", "WinMX", "eMule", "WASTE", "Kazaa", "Direct"

**Sites:** "Yahoo", "Amazon", "Microsoft", "Bing", "MSN", "Google", "YouTube", "Facebook", "Twitter"