visualization

December 19, 2024

1 Visualization

Nov 10th / Yuqi

delays.tbss (TBS) delays.mcss (MCS) delays.segments (number of segmentations)

ideal causal relationship: TBS –> number of segements <– MCS

csv_file panda_frame

1.1 1 Import

[1]:
```
from plot_helpers import *
from data_helpers import *
from decomp import *

import os, sys, gzip, json
from pathlib import Path
from sortedcontainers import SortedList, SortedDict
from loguru import logger
import pandas as pd
import sqlite3
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as patches
import numpy as np
from IPython.display import JSON
import ijson
import seaborn as sns
import pandas as pd

%load_ext autoreload
%autoreload 2

sns.set_theme(style='darkgrid')  # Options include 'darkgrid', 'whitegrid',␣
 ↪'dark', 'white', and 'ticks'
sns.set()
```

[2]:
```
# Remove default handler
logger.remove()
# Add a new handler with level WARNING
logger.add(sys.stdout, level="ERROR")
```

[2]:
```
2
```

1.2 2 Initalize paths

[3]:
```
DATASETS_DIR = "./data/"
PLOTS_DIR = "./plots/"
JSON_PATH = "jsons/"
IF_SHOW_USAGE = True
```

1.3 3 class Meas for visualizing single dataset

This block aims to construct a clear and consice way of visualization by code encapsulation.

[4]:
```
SKIP_FIRST = 1000
SKIP_LAST = 100
class Meas:
    class Data:
        def __init__(self, datasets_dir,meas_label):
            """
            Meas.Data: Store the data from one measurement
            """
            self.meas_label = meas_label
            self.data_path = datasets_dir + meas_label
            jsons_path = os.path.join(self.data_path, JSON_PATH)
            self.packets = read_json(os.path.join(jsons_path, "packets.json"))
            self.sr_tx_arr = read_json(os.path.join(jsons_path, "sr_tx.json"))
            self.bsr_tx_arr = read_json(os.path.join(jsons_path, "bsr_tx.json"))
            self.bsrupd_arr = read_json(os.path.join(jsons_path, "bsr_upd.
 ↪json"))
            self.bsrupd_sorted_dict = SortedDict(
                {bsrupd["timestamp"]: bsrupd for bsrupd in self.bsrupd_arr}
            )
            self.sr_bsr_tx_sorted_list = SortedList(
                [sr_tx["timestamp"] for sr_tx in self.sr_tx_arr]
                + [bsr_tx["timestamp"] for bsr_tx in self.bsr_tx_arr]
            )
            self.sched_arr = read_json(os.path.join(jsons_path, "sched.json"))
            self.sched_decid_sorted_dict = SortedDict(
                {sched["decision_ts"]: sched for sched in self.sched_arr}
            )
            self.sched_sched_sorted_dict = SortedDict(
                {sched["schedule_ts"]: sched for sched in self.sched_arr}
            )
            self.packets_rnti_set = set(  # Radio Network Temporary Identifiers
                [
                    item["rlc.attempts"][0]["rnti"]
                    for item in self.packets
                    if item["rlc.attempts"][0]["rnti"] != None
                ]
            )
            print(f"RNTIs in packets of {self.meas_label}: {list(self.
    ↪packets_rnti_set)}")
            self.mcs_arr_all = read_json(os.path.join(jsons_path, "mcs.json"))
            self.tb_arr = read_json(os.path.join(jsons_path, "tb.json"))
            self.set_rnti = set([item["rnti"] for item in self.mcs_arr_all])
            # filter entries with rnti list(packets_rnti_set)[0]
            if list(self.packets_rnti_set)[0] != None:
                self.mcs_arr = [
                    mcs
                    for mcs in self.mcs_arr_all
                    if mcs["rnti"] == list(self.packets_rnti_set)[0]
                ]
            else:
                self.mcs_arr = self.mcs_arr_all
            self.mcs_sorted_dict = SortedDict(
                {mcs["timestamp"]: mcs for mcs in self.mcs_arr}
            )

    class Delays:
        def __init__(self, data):
            """
            Meas.Delays: Calculate and store delay components, utilizing decomp.
    ↪py
            """

            self.tbss = []
            last_valid_tbs = None  # Initialize last_valid_tbs outside the loop

            for packet in data.packets:
                tbs = get_tbs(
                    packet,
                    data.sched_decid_sorted_dict,
                    data.sched_sched_sorted_dict,
                    slots_duration_ms=0.5,
                )
                tx_delay = get_tx_delay(packet)
                if tbs is not None and tx_delay is not None:
                    last_valid_tbs = tbs
                    tbs_val = last_valid_tbs
                elif tx_delay is not None:
                    tbs_val = last_valid_tbs
                else:
                    tbs_val = None

                self.tbss.append(tbs_val)

                if tbs_val == None and get_segments(packet) != None and␣
↪get_tx_delay(packet) !=None:
                    print(f"For packet with {packet["id"]} in {data.
↪meas_label}, tbs is None but segments is not None, Remove this packet!")
                    the_id = packet["id"]
                    data.packets = [
                        packet for packet in data.packets if packet.get("id") !
↪= the_id
                    ]

            self.tbss = np.array(
                [
                    item
                    for item in self.tbss
                    if item != None
                ]
            )

            self.idt = np.array(
                list(
                    {
                        data.packets[ind]["id"]: data.packets[ind]["ip.in_t"]
                        - data.packets[ind - 1]["ip.in_t"]
                        for ind in range(1, len(data.packets))
                        if data.packets[ind]["ip.in_t"] != None
                        and data.packets[ind - 1]["ip.in_t"] != None
                    }.values()
                )
            )

            self.frame_alignment_delays = np.array(
                list(
                    {
                        packet["id"]: get_frame_alignment_delay(
                            packet,
                            data.sr_bsr_tx_sorted_list,
                            slots_per_frame=20,
                            slots_duration_ms=0.5,
                        )
                        for packet in data.packets
                        if get_frame_alignment_delay(
                            packet,
                            data.sr_bsr_tx_sorted_list,
                            slots_per_frame=20,
                            slots_duration_ms=0.5,
                        )
                        != None
                        and get_tx_delay(packet) != None
                    }.values()
                )
            )
            self.scheduling_delays = np.array(
                list(
                    {
                        packet["id"]: get_scheduling_delay(
                            packet,
                            data.sched_decid_sorted_dict,
                            data.sched_sched_sorted_dict,
                            slots_per_frame=20,
                            slots_duration_ms=0.5,
                        )
                        for packet in data.packets
                        if get_scheduling_delay(
                            packet,
                            data.sched_decid_sorted_dict,
                            data.sched_sched_sorted_dict,
                            slots_per_frame=20,
                            slots_duration_ms=0.5,
                        )
                        != None
                        and get_tx_delay(packet) != None
                    }.values()
                )
            )
            self.ran_delays = np.array(
                list(
                    {
                        packet["id"]: get_ran_delay(packet)
                        for packet in data.packets
                        if get_ran_delay(packet) != None
                    }.values()
                )
            )
            self.ran_delays_wo_frame_alignment_delay = np.array(
                list(
                    {
                        packet["id"]: get_ran_delay_wo_frame_alignment_delay(
                            packet,
                            data.sr_bsr_tx_sorted_list,
                            slots_per_frame=20,
                            slots_duration_ms=0.5,
                        )
                        for packet in data.packets
                        if get_ran_delay_wo_frame_alignment_delay(
                            packet,
                            data.sr_bsr_tx_sorted_list,
                            slots_per_frame=20,
                            slots_duration_ms=0.5,
                        )
                        != None
                    }.values()
                )
            )
            self.ran_delays_wo_scheduling_delay = np.array(
                list(
                    {
                        packet["id"]: get_ran_delay_wo_scheduling_delay(
                            packet,
                            data.sched_decid_sorted_dict,
                            data.sched_sched_sorted_dict,
                            slots_per_frame=20,
                            slots_duration_ms=0.5,
                        )
                        for packet in data.packets
                        if get_ran_delay_wo_scheduling_delay(
                            packet,
                            data.sched_decid_sorted_dict,
                            data.sched_sched_sorted_dict,
                            slots_per_frame=20,
                            slots_duration_ms=0.5,
                        )
                        != None
                    }.values()
                )
            )
            self.queueing_delays = np.array(
                list(
                    {
                        packet["id"]: get_queueing_delay(packet)
                        for packet in data.packets
                        if get_queueing_delay(packet) != None and␣
    ↪get_tx_delay(packet) !=None
                    }.values()
                )
            )
            self.queueing_delays_wo_scheduling_delay = np.array(
                list(
                    {
                        packet["id"]: get_queueing_delay_wo_scheduling_delay(
                            packet,
                            data.sched_decid_sorted_dict,
                            data.sched_sched_sorted_dict,
                            slots_per_frame=20,
                            slots_duration_ms=0.5,
                        )
                        for packet in data.packets
                        if get_queueing_delay_wo_scheduling_delay(
                            packet,
                            data.sched_decid_sorted_dict,
                            data.sched_sched_sorted_dict,
                            slots_per_frame=20,
                            slots_duration_ms=0.5,
                        )
                        != None
                        and get_tx_delay(packet) != None
                    }.values()
                )
            )
            self.segmentation_delay = np.array(
                list(
                    {
                        packet["id"]: get_segmentation_delay(packet)
                        for packet in data.packets
                        if get_segmentation_delay(packet) != None
                    }.values()
                )
            )
            self.segmentation_delays_wo_scheduling_delay = np.array(
                list(
                    {
                        packet["id"]:␣
↪get_segmentation_delay_wo_scheduling_delay(
                            packet,
                            data.sched_decid_sorted_dict,
                            data.sched_sched_sorted_dict,
                            slots_per_frame=20,
                            slots_duration_ms=0.5,
                        )
                        for packet in data.packets
                        if get_segmentation_delay_wo_scheduling_delay(
                            packet,
                            data.sched_decid_sorted_dict,
                            data.sched_sched_sorted_dict,
                            slots_per_frame=20,
                            slots_duration_ms=0.5,
                        )
                        != None
                    }.values()
                )
            )

            # This is commented, because frame_alignment_delay is part of the␣
↪scheduling_delay!
            # self.segmentation_delays_wo_scheduling_framealignment_delay =␣
↪self.segmentation_delays_wo_scheduling_delay - self.frame_alignment_delays

            self.segments = np.array(
                list(
                    {
                        packet["id"]: get_segments(packet)
                        for packet in data.packets
                        if get_segments(packet) != None and␣
↪get_tx_delay(packet) !=None
                    }.values()
                )
            )
            self.retx_delays = np.array(
                list(
                    {
                        packet["id"]: get_retx_delay(packet)
                        for packet in data.packets
                        if get_retx_delay(packet) != None
                    }.values()
                )
            )
            self.mcss=np.array(
                list(
                    {
                        packet["id"]: get_mcs(
                            packet,
                            data.mcs_sorted_dict,
                            slots_per_frame=20,
                            slots_duration_ms=0.5,
                        )
                        for packet in data.packets
                        if get_mcs(
                            packet,
                            data.mcs_sorted_dict,
                            slots_per_frame=20,
                            slots_duration_ms=0.5,
                        )
                        != None and get_tx_delay(packet) != None
                    }.values()
                )
            )
            # self.tbss = np.array(
            #     list(
            #         {
            #             packet["id"]: get_tbs(
            #                 packet, data.sched_sorted_dict,␣
↪slots_duration_ms=0.5
            #             )
            #             for packet in data.packets
            #             if get_tbs(
            #                 packet, data.sched_sorted_dict,␣
↪slots_duration_ms=0.5
            #             )
            #             != None
            #         }.values()
            #     )
            # )

            # Here, if "tbs" not exist, we assume the the tbs for this packet␣
↪remains unchanged
            # last_valid_tbs = None  # Initialize to keep track of the last␣
↪valid TBS value
            # self.tbss = np.array(
            #     [
            #         # Iterate over each packet to fetch TBS or use last valid␣
↪TBS if None
            #         (
            #             (
            #                 last_valid_tbs := get_tbs(
            #                     packet,
            #                     data.sched_decid_sorted_dict,
            #                     data.sched_sched_sorted_dict,
            #                     slots_duration_ms=0.5,
            #                 )
            #             )
            #             if (
            #                 tbs := get_tbs(
            #                     packet,
            #                     data.sched_decid_sorted_dict,
            #                     data.sched_sched_sorted_dict,
            #                     slots_duration_ms=0.5,
            #               )
            #           )
            #           is not None and get_tx_delay(packet) != None
            #           else last_valid_tbs
            #                if get_tx_delay(packet) != None
            #                else None
            #       )
            #       for packet in data.packets
            #   ]
            # )

            self.packet_size = np.array(
                list(
                    {
                        packet["id"]: get_packet_size(packet)
                        for packet in data.packets
                        if get_packet_size(packet) != None
                        and get_tx_delay(packet) != None
                    }.values()
                )
            )
            self.timestamps = np.array(
                list(
                    {
                        packet["id"]: packet["ip.in_t"]
                        for packet in data.packets
                        if packet["ip.in_t"] != None and get_tx_delay(packet) !
↪= None
                    }
                )
            )

    def __init__(self, datasets_dir=DATASETS_DIR, meas_label="s49"):
        """
        Meas:
            a class which store and analyze 1 group(1 folder) of measurement.␣
↪The init function utilizes data_helps.py
        Parameters:
            datasets_dir(str): Path of all datasets;
            meas_label(str): The measurement label;
        """
        self.meas_label=meas_label
        self.data=self.Data(datasets_dir, meas_label)
        self.delays=self.Delays(self.data)
        if not os.path.exists(PLOTS_DIR + self.data.meas_label):
            os.makedirs(PLOTS_DIR + self.data.meas_label)
    def checkData(self, name, pkt_idx=0):
        """
        checkData: check the 1st json object of given attribute name

        Parameters:
            name(str): the attribute name of Meas.Delay\
            pkt_idx(int): the index of checked json object
        """
        attr=getattr(self.data, name, None)
        if attr is not None:
            for idx,attr_item in enumerate(attr):
                if idx <pkt_idx:
                    continue
                elif idx == pkt_idx:
                    print(json.dumps(attr_item, indent=4, allow_nan=True))
                else:
                    break
        else:
            print(f"No attribute {name} found in Meas_{self.data.meas_label}.
↪data")

    def listDataAttr(self):
        return list(vars(self.data).keys()) # ['attr1', 'attr2']

    def listDelaysAttr(self):
        return list(vars(self.delays).keys())

    def plotCCDF(self, curve_name,skip_first=SKIP_FIRST, skip_last=SKIP_LAST,␣
↪figsize=(8,5), plt_show=True, ax_external=None, outlier=35, x_lim=30):
        """
        Meas.plotCCDF:
            plot 1 ccdf among 13 kinds of delay measurement

        Parameters:
            curve_name(str):the delay measurement you want to plot
            skip_first(int): skip first a few packets
            skip_last(int):  skip last a few packets
            figsize(tuple):  figsize
            plt_show(bool):  if or not show and save figure.
            ax_external(object): external plot object
        """
        curve=getattr(self.delays, curve_name, None)
        if curve is not None:
            if curve_name == "packet_size":
                outlier = None
                x_lim=512
            plot_ccdf(curve[skip_first:-skip_last], f"{self.data.
↪meas_label}_{curve_name}", figsize=figsize, ax=ax_external, outlier=outlier,␣
↪x_lim = x_lim)
            if plt_show:
                plt.tight_layout()
                plt.savefig(f"{PLOTS_DIR}{self.data.meas_label}/
↪{curve_name}_ccdf_plot.png", dpi=300, bbox_inches='tight')
                plt.show()
        else:
            print(f"No attribute {curve_name} found in meas.delays of {self.
↪data.meas_label}.")

    def plotAllCCDF(self, figsize=(24,24),subplot_division=[1,1]):
        """
        Meas.plotAllCCDF:
            plot ccdf for 13 kinds of delay measurement

        Parameters:
            figsize(tuple):  figsize
            subplot_division(list=[x,y]): x-row y-column subplots. If [1,1],␣
↪figures will be plotted separately.
        """
        curve_names = self.listDelaysAttr()
        curve_names = [curve_name for curve_name in curve_names if curve_name !
↪= 'timestamps']
        figure_num=0
        for i in range(0,len(curve_names)):
            if i % (subplot_division[0]*subplot_division[1]) == 0:
                if i != 0:
                    plt.tight_layout()
                    plt.savefig(f"{PLOTS_DIR}{self.data.meas_label}/
↪all_ccdf_plots_{figure_num}.png", dpi=300)
                    figure_num=figure_num+1
                    plt.show()
                _, axs = plt.subplots(subplot_division[0], subplot_division[1],␣
↪figsize=figsize)

            if subplot_division[0]*subplot_division[1]==1:
                self.plotCCDF(curve_names[i], plt_show=False, ax_external=axs)
            elif subplot_division[0]==1 or subplot_division[1]==1:
                self.plotCCDF(curve_names[i], plt_show=False, ax_external=axs[i␣
↪% (subplot_division[1]*subplot_division[0])])
            else:
                self.plotCCDF(
                    curve_names[i],
                    plt_show=False,
                    ax_external=axs[
                        (i % (subplot_division[1]*subplot_division[0] ) //␣
↪subplot_division[1]),
                        i % subplot_division[1],
                    ],
                )
            plt.tight_layout()
            plt.savefig(f"{PLOTS_DIR}{self.data.meas_label}/
↪all_ccdf_plots_{figure_num}.png", dpi=300, bbox_inches="tight")
            plt.show()

    def plotTimeSeries(self, curve_names, curves=None, start_idx=5000,␣
↪end_idx=6000, figsize=(12,5), marker="*", title="timeseries",plt_show=True,␣
↪ax_external=None):
        if ax_external is not None:
            ax = ax_external
        else:
            _, ax = plt.subplots(figsize=figsize)
        index_range = slice(start_idx, end_idx)
        for idx,curve_name in enumerate(curve_names):
            if curves is not None:
                ax.plot(curves[idx][index_range], marker=marker, label=f"{self.
↪data.meas_label}: {curve_name}")
            else:
                curve = getattr(self.delays, curve_name, None)
                if curve is not None:
                    ax.plot(curve[index_range], marker=marker, label=f"{self.
↪data.meas_label}: {curve_name}")
                else:
                    print(
                        f"Curve {curve_name} not found in meas.delays of {self.
↪data.meas_label}."
                    )
        ax.set_xlabel("Index")
        ax.set_ylabel("Delay [ms]")
        ax.grid(True,"minor")
        ax.legend()
        if plt_show == True:
            i = 1
            while os.path.exists(os.path.join(PLOTS_DIR, self.data.meas_label,␣
↪f"{title}_{i}.png")):
                i += 1
            plt.tight_layout()
            plt.savefig(
                f"{PLOTS_DIR}{self.data.meas_label}/{title}_{i}.png", dpi=300,␣
↪bbox_inches="tight"
```

```
            )
            plt.show()

    def dataFrame(self, curve_names, curve_labels=None, csv_path=None,␣
↪if_save=True):
        """
        Export dataframe of given delay components with given labels, and save␣
↪it to csv file.

        curve_labels(list of str): data labels
        curve_names(list of str): attributes names of Meas.Delay (use Meas.
↪listDelaysAttr() to check)
        csv_path(str): Optional. default: the data folder of the meas.
        """
        data=dict()
        for idx, curve_name in enumerate(curve_names):
            curve = getattr(self.delays, curve_name, None)
            if curve is not None:
                if curve_labels is not None:
                    data[curve_labels[idx]]=curve
                else:
                    data[curve_names[idx]] = curve

        # Create a pandas DataFrame with each array as a column
        df = pd.DataFrame(
            data
        )
        # Display the first few rows of the DataFrame
        print(df.head())

        # Export the DataFrame to a CSV file
        if csv_path == None:
            csv_path=f"{DATASETS_DIR}{self.data.meas_label}"
            print(csv_path)

        if if_save ==True:
            df.to_csv(os.path.join(csv_path, f"{"_".join(curve_labels)}.csv"),␣
↪index=True)
            print(f"Dataframe saved to {os.path.join(csv_path, f"{"_".
↪join(curve_labels)}.csv")}")

        return df

    def plotCrossCorrelation(self,  curve_names, curve_labels= None):
        """
        Calculates and returns the Pearson correlation coefficient between two␣
↪vectors.
```

```
        curve_names(list of str): attributes names of Meas.Delay (use Meas.
↪listDelaysAttr() to check)
        curve_labels(list of str): Optional,data labels
        """
        df = self.dataFrame(curve_names, curve_labels=curve_labels,␣
↪if_save=False)

        fig = plt.figure()
        ax = fig.add_subplot(111)
        correlation_matrix = df.corr()
        sns.heatmap(df.corr(), ax=ax, annot=True)
        plt.show()

        return correlation_matrix

    def plotAutoCorrelation(self, curve_name, label=None, figsize=(8,5),␣
↪plt_show=True, ax_external=None, outlier = 35):
        """
        Meas.plotCCDF:
            plot 1 autocorr among all kinds of delay measurement

        Parameters:
            curve_name(str):the delay measurement you want to plot
            figsize(tuple):  figsize
            plt_show(bool):  if or not show and save figure.
            ax_external(object): external plot object
        """
        curve=getattr(self.delays, curve_name, None)
        if label is None:
            label = f"{self.data.meas_label}_{curve_name}"
        if curve is not None:
            plot_autocorr(
                curve, label, figsize=figsize, ax=ax_external, outlier=outlier
            )
            if plt_show:
                plt.tight_layout()
                plt.savefig(f"{PLOTS_DIR}{self.data.meas_label}/
↪{curve_name}_autocorr_plot.png", dpi=300, bbox_inches='tight')
                plt.show()
        else:
            print(f"No attribute {curve_name} found in meas.delays of {self.
↪data.meas_label}.")

    def plotAllAutoCorrelation(self, delays=None, labels = None,␣
↪figsize=(24,24), subplot_division=[1,1], outlier=35, x_lim=100 ):
        if delays == None:
            curve_names = self.listDelaysAttr()
        else:
            curve_names = delays
        if labels == None:
            labels = curve_names
        figure_num = 0
        for i in range(0, len(curve_names)):
            if i % (subplot_division[0] * subplot_division[1]) == 0:
                if i != 0:
                    plt.tight_layout()
                    plt.savefig(
                        f"{PLOTS_DIR}{self.data.meas_label}/
↪AutoCorrelation_{figure_num}.png",
                        dpi=300,
                    )
                    figure_num = figure_num + 1
                    plt.show()
                _, axs = plt.subplots(
                    subplot_division[0], subplot_division[1], figsize=figsize
                )

            if subplot_division[0] * subplot_division[1] == 1:
                self.plotAutoCorrelation(curve_names[i],␣
↪label=labels[i],plt_show=False, ax_external=axs, outlier=outlier)
            elif subplot_division[0] == 1 or subplot_division[1] == 1:
                self.plotAutoCorrelation(
                    curve_names[i],
                    label=labels[i],
                    plt_show=False,
                    ax_external=axs[i % (subplot_division[1] *␣
↪subplot_division[0])],
                    outlier=outlier,
                )
            else:
                self.plotAutoCorrelation(
                    curve_names[i],
                    label=labels[i],
                    plt_show=False,
                    ax_external=axs[
                        (
                            i
                            % (subplot_division[1] * subplot_division[0])
                            // subplot_division[1]
                        ),
                        i % subplot_division[1],
                    ],
                    outlier=outlier,
                )
        plt.tight_layout()
        plt.savefig(
            f"{PLOTS_DIR}{self.data.meas_label}/AutoCorrelation_{figure_num}.
↪png",
            dpi=300,
            bbox_inches="tight",
        )
        plt.show()

    def crosscorr(self, datax, datay, lag=0, wrap=False):
        """Lag-N cross correlation with time-lagging capability.

        Parameters
        ----------
        datax, datay : pandas.Series
            The time series data to compute the cross-correlation.
        lag : int, default 0
            The lag (positive or negative) for the cross-correlation.
        wrap : bool, default False
            If True, wraps the lagged values (useful for cyclic data).

        Returns
        ----------
        crosscorr : float
            The Pearson correlation coefficient for the given lag.
        """
        if wrap:
            shiftedy = datay.copy()  # Create a copy to modify
            if lag > 0:
                # Wrap last `lag` values to the beginning
                shiftedy.iloc[:lag] = datay.iloc[-lag:].values
            elif lag < 0:
                # Wrap first `-lag` values to the end
                shiftedy.iloc[lag:] = datay.iloc[:lag + len(datay)].values
            # Compute correlation
            return datax.corr(shiftedy)
        else:
            # Standard shift with NaN filling
            return datax.corr(datay.shift(lag))

    def plotTLCC(self, curve_names, curve_labels=None, wrap = False):
        data = dict()
        if curve_labels is None:
            curve_labels = curve_names
        for idx, curve_name in enumerate(curve_names):
            curve = getattr(self.delays, curve_name, None)
            if curve is not None:
                data[curve_labels[idx]] = curve
            else:
                print(f"{curve_name} not found")
                return

        # Create a pandas DataFrame with each array as a column
        df = pd.DataFrame(data)
        d1 = df[curve_labels[0]]
        d2 = df[curve_labels[1]]

        max_lag = 50
        lags = range(-max_lag, max_lag + 1)

        rs = [self.crosscorr(d1, d2, lag, wrap=wrap) for lag in lags]

        peak_lag = lags[np.argmax(np.abs(rs))]
        d2_offset = -peak_lag

        f, ax = plt.subplots(figsize=(14, 3))
        ax.plot(lags, rs, label="Cross-correlation")
        ax.axvline(0, color='k', linestyle='--', label='Center (Lag=0)')
        ax.axvline(peak_lag, color='r', linestyle='--', label='Peak Synchrony')
        ax.set(
            title=(
                f"{curve_labels[0]}[i-{d2_offset}] ~ {curve_labels[1]}[i]"
                if d2_offset > 0
                else (
                    f"{curve_labels[0]}[i] ~ {curve_labels[1]}[i{d2_offset}]"
                    if d2_offset < 0
                    else f"{curve_labels[0]}[i] ~ {curve_labels[1]}[i]"
                )
            ),
            ylim=[-1, 1],
            xlabel="Lag (packets)",
            ylabel="Pearson r",
        )
        ax.legend()
        plt.show()
        return
```

1.3.1 3.1 Usage of class Meas

3.1.1 Import dataset

```
[5]: if IF_SHOW_USAGE == True:
         Meas_s54=Meas(meas_label='s54')
```

```
RNTIs in packets of s54: ['b4f7']
2024-12-19 15:54:18.546 | ERROR    |
decomp:get_tx_delay:246 - Packet 37888

phy.in_t or phy.in_t not present
2024-12-19 15:54:18.547 | ERROR    |
decomp:get_tx_delay:246 - Packet 37887

phy.in_t or phy.in_t not present
2024-12-19 15:54:18.633 | ERROR    |
decomp:get_tx_delay:246 - Packet 33978

phy.in_t or phy.in_t not present
2024-12-19 15:54:18.634 | ERROR    |
decomp:get_tx_delay:246 - Packet 33977

phy.in_t or phy.in_t not present
2024-12-19 15:54:18.714 | ERROR    |
decomp:get_tx_delay:246 - Packet 30349

phy.in_t or phy.in_t not present
2024-12-19 15:54:18.721 | ERROR    |
decomp:get_tx_delay:246 - Packet 30028

phy.in_t or phy.in_t not present
2024-12-19 15:54:18.745 | ERROR    |
decomp:get_tx_delay:246 - Packet 29033

phy.in_t or phy.in_t not present
2024-12-19 15:54:18.869 | ERROR    |
decomp:get_tx_delay:246 - Packet 23970

phy.in_t or phy.in_t not present
2024-12-19 15:54:18.870 | ERROR    |
decomp:get_tx_delay:246 - Packet 23969

phy.in_t or phy.in_t not present
2024-12-19 15:54:18.948 | ERROR    |
decomp:get_tx_delay:246 - Packet 19965

phy.in_t or phy.in_t not present
2024-12-19 15:54:18.949 | ERROR    |
decomp:get_tx_delay:246 - Packet 19964

phy.in_t or phy.in_t not present
2024-12-19 15:54:18.991 | ERROR    |
decomp:get_tx_delay:246 - Packet 17962

phy.in_t or phy.in_t not present
2024-12-19 15:54:18.992 | ERROR    |
decomp:get_tx_delay:246 - Packet 17961

phy.in_t or phy.in_t not present
```

```
decomp:get_tx_delay:246 - Packet 8707

phy.in_t or phy.in_t not present
2024-12-19 15:54:36.443 | ERROR    |
decomp:get_tx_delay:246 - Packet 8544

phy.in_t or phy.in_t not present
2024-12-19 15:54:36.472 | ERROR    |
decomp:get_tx_delay:246 - Packet 5948

phy.in_t or phy.in_t not present
2024-12-19 15:54:36.474 | ERROR    |
decomp:get_tx_delay:246 - Packet 5947

phy.in_t or phy.in_t not present
2024-12-19 15:54:36.483 | ERROR    |
decomp:get_tx_delay:246 - Packet 5393

phy.in_t or phy.in_t not present
2024-12-19 15:54:36.486 | ERROR    |
decomp:get_tx_delay:246 - Packet 4974

phy.in_t or phy.in_t not present
2024-12-19 15:54:36.490 | ERROR    |
decomp:get_tx_delay:246 - Packet 4973

phy.in_t or phy.in_t not present
2024-12-19 15:54:36.500 | ERROR    |
decomp:get_tx_delay:246 - Packet 3944

phy.in_t or phy.in_t not present
2024-12-19 15:54:36.502 | ERROR    |
decomp:get_tx_delay:246 - Packet 3943

phy.in_t or phy.in_t not present
2024-12-19 15:54:36.546 | ERROR    |
decomp:get_tx_delay:246 - Packet 2168

phy.in_t or phy.in_t not present
2024-12-19 15:54:36.557 | ERROR    |
decomp:get_tx_delay:246 - Packet 1941

phy.in_t or phy.in_t not present
2024-12-19 15:54:36.559 | ERROR    |
decomp:get_tx_delay:246 - Packet 1940

phy.in_t or phy.in_t not present
2024-12-19 15:54:36.578 | ERROR    |
decomp:get_tx_delay:246 - Packet 1098

phy.in_t or phy.in_t not present
```

Some test

[6]:
```
#Meas_s49.data.sched_sorted_dict[Meas_s49.data.sched_sorted_dict.
 ↪keys()[1123]]["cause"]["tbs"]
# Meas_s49.checkData("mcs_sorted_dict")
# Meas_s49.checkData("packets")
```

[7]:
```
if IF_SHOW_USAGE== True:
    print(list(vars(Meas_s54.delays).keys()))  # ['attr1', 'attr2']

    for attr_name in list(vars(Meas_s54.delays).keys()):
        attr = getattr(Meas_s54.delays, attr_name, None)
        if attr is not None:
            print(
                f"len(Meas_{Meas_s54.data.meas_label}.delays.{attr_name})=␣
 ↪{len(attr)} "
            )
```

```
['tbss', 'idt', 'frame_alignment_delays', 'scheduling_delays', 'ran_delays',
'ran_delays_wo_frame_alignment_delay', 'ran_delays_wo_scheduling_delay',
'queueing_delays', 'queueing_delays_wo_scheduling_delay', 'segmentation_delay',
'segmentation_delays_wo_scheduling_delay', 'segments', 'retx_delays', 'mcss',
'packet_size', 'timestamps']
len(Meas_s54.delays.tbss)= 39999
len(Meas_s54.delays.idt)= 39998
len(Meas_s54.delays.frame_alignment_delays)= 39999
len(Meas_s54.delays.scheduling_delays)= 39999
len(Meas_s54.delays.ran_delays)= 39999
len(Meas_s54.delays.ran_delays_wo_frame_alignment_delay)= 39999
len(Meas_s54.delays.ran_delays_wo_scheduling_delay)= 39999
len(Meas_s54.delays.queueing_delays)= 39999
len(Meas_s54.delays.queueing_delays_wo_scheduling_delay)= 39999
len(Meas_s54.delays.segmentation_delay)= 39999
len(Meas_s54.delays.segmentation_delays_wo_scheduling_delay)= 39998
len(Meas_s54.delays.segments)= 39999
len(Meas_s54.delays.retx_delays)= 39999
len(Meas_s54.delays.mcss)= 39999
len(Meas_s54.delays.packet_size)= 39999
len(Meas_s54.delays.timestamps)= 39999
```

[8]:
```
if IF_SHOW_USAGE == True:
    print(type(Meas_s54.delays.tbss))
    print(Meas_s54.delays.tbss.shape)
```

```
<class 'numpy.ndarray'>
(39999,)
```

3.1.2 `checkData`, `listDataAttr`, `listDelaysAttr`

`checkData`: print the [pkt__idx]-th json object in the given Meas.Data attribute name

`listDataAttr`: print all attributes names in the given Meas.Data `listDelaysAttr` : print all attributes names in the given Meas.Delays

Here are all attributes retrieved from json files, stored in [Meas_instance].data

```
[9]: if IF_SHOW_USAGE == True:
         Meas_s54.listDataAttr()
```

```
[10]: if IF_SHOW_USAGE == True:
          Meas_s54.listDelaysAttr()
```

```
[11]: if IF_SHOW_USAGE == True:
          Meas_s54.checkData("sched_arr")
```

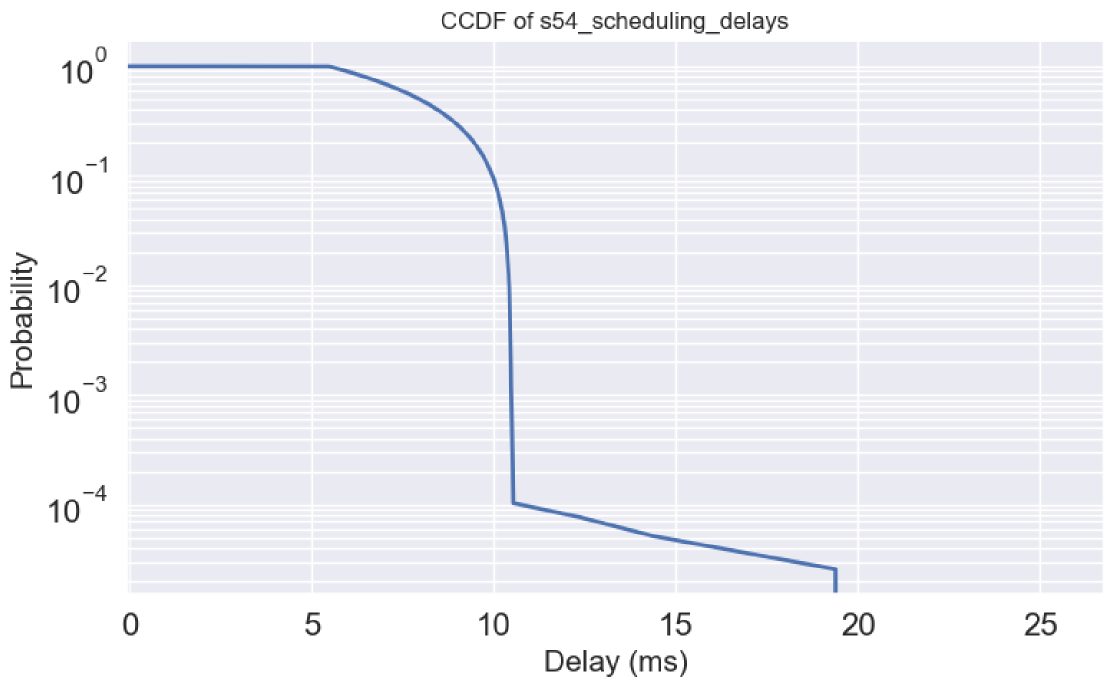
```
{
    "decision_ts": 1730654462.927429,
    "schedule_ts": 1730654462.930429,
    "symbols_start": 10,
    "symbols_num": 3,
    "prbs_start": 0,
    "prbs_num": 5,
    "cause": {
        "rnti": "b4f7",
        "tbs": 24,
        "mcs": 9,
        "rbs": 5,
        "type": 3,
        "diff": 7124.0,
        "buf": NaN,
        "sched": NaN,
        "hqround": NaN,
        "hqpid": NaN
    }
}
```

3.1.3 `plotCCDF` plot CCDF of single delay component

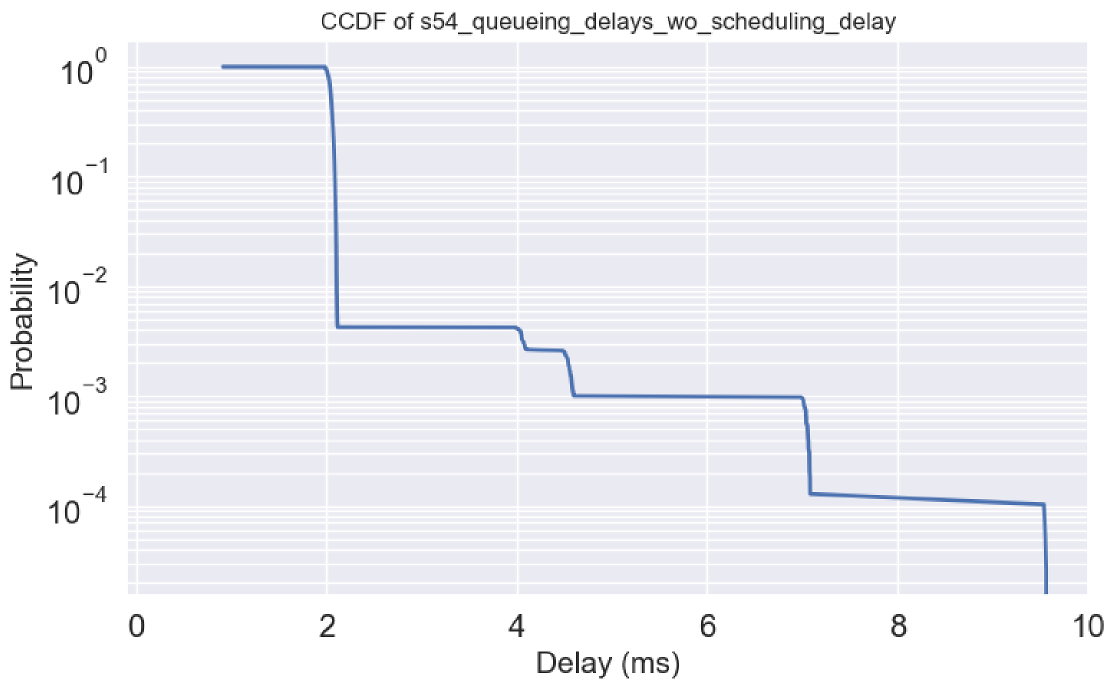
```
[12]: if IF_SHOW_USAGE == True:
          print(list(vars(Meas_s54.delays).keys()))  # ['attr1', 'attr2']
```

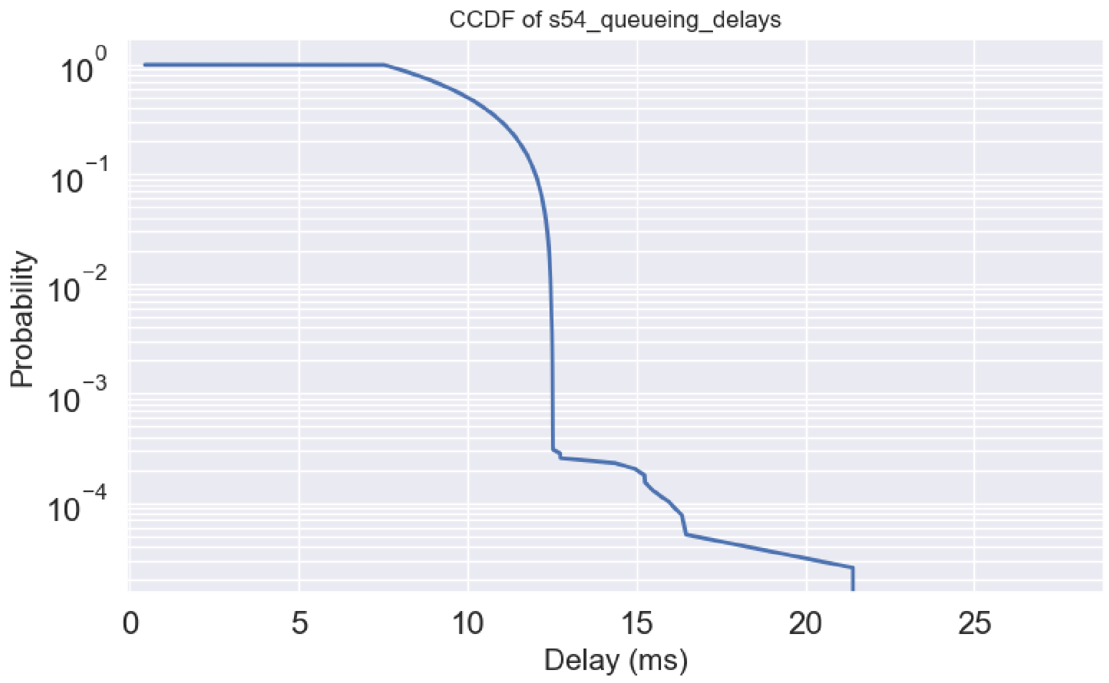
```
['tbss', 'idt', 'frame_alignment_delays', 'scheduling_delays', 'ran_delays',
'ran_delays_wo_frame_alignment_delay', 'ran_delays_wo_scheduling_delay',
'queueing_delays', 'queueing_delays_wo_scheduling_delay', 'segmentation_delay',
'segmentation_delays_wo_scheduling_delay', 'segments', 'retx_delays', 'mcss',
'packet_size', 'timestamps']
```

```
[13]: if IF_SHOW_USAGE == True:
          Meas_s54.plotCCDF("scheduling_delays")
```



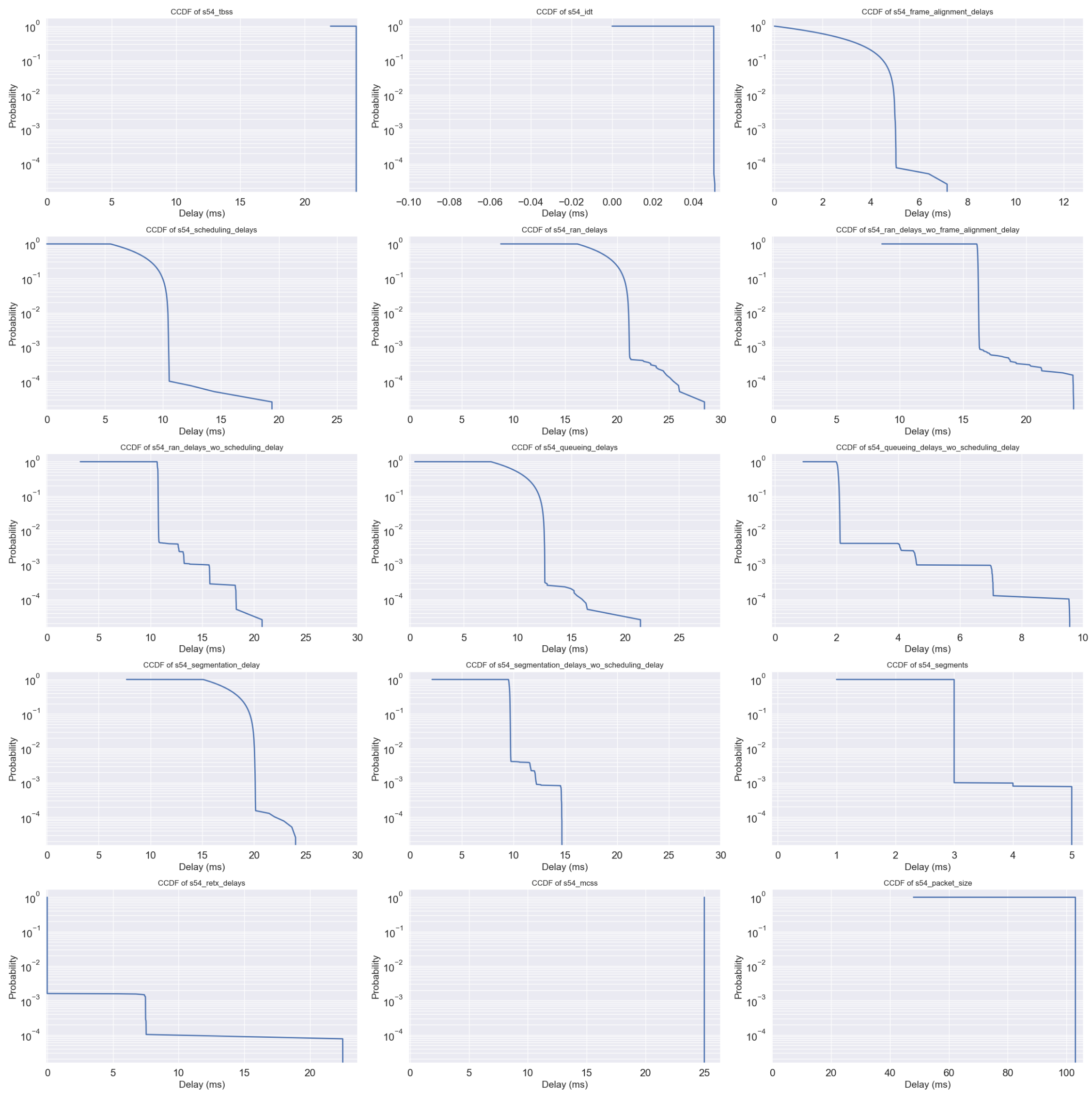
```
[14]: if IF_SHOW_USAGE == True:
          Meas_s54.plotCCDF("queueing_delays_wo_scheduling_delay")
          Meas_s54.plotCCDF("queueing_delays")
```





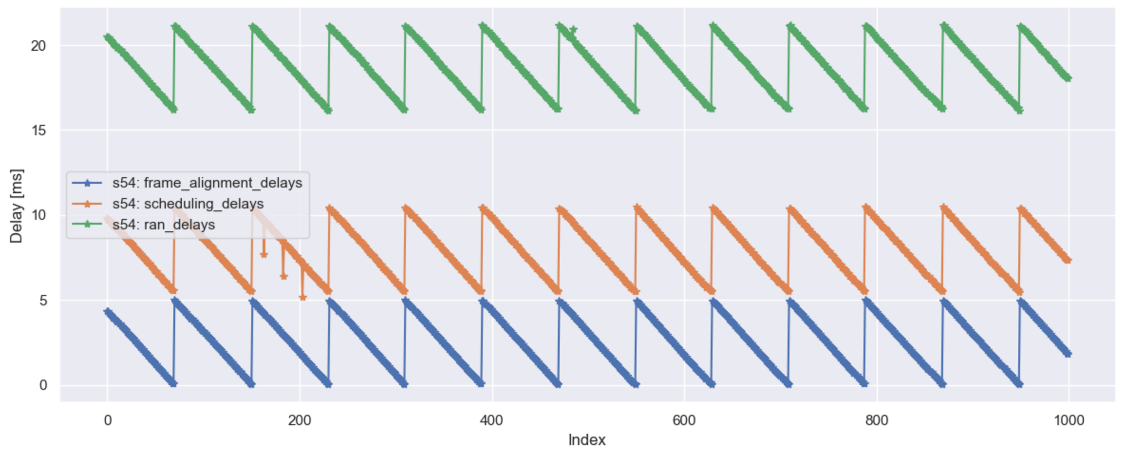
3.1.4 `plotAllCCDF` plot all CCDF curves

[15]:
```
if IF_SHOW_USAGE == True:
    Meas_s54.plotAllCCDF(subplot_division=[5, 3])

# Meas_s49.plotAllCCDF(subplot_division=[1, 1])

# Meas_s49.plotAllCCDF(subplot_division=[1, 3])

# Meas_s49.plotAllCCDF(subplot_division=[3, 1])
```

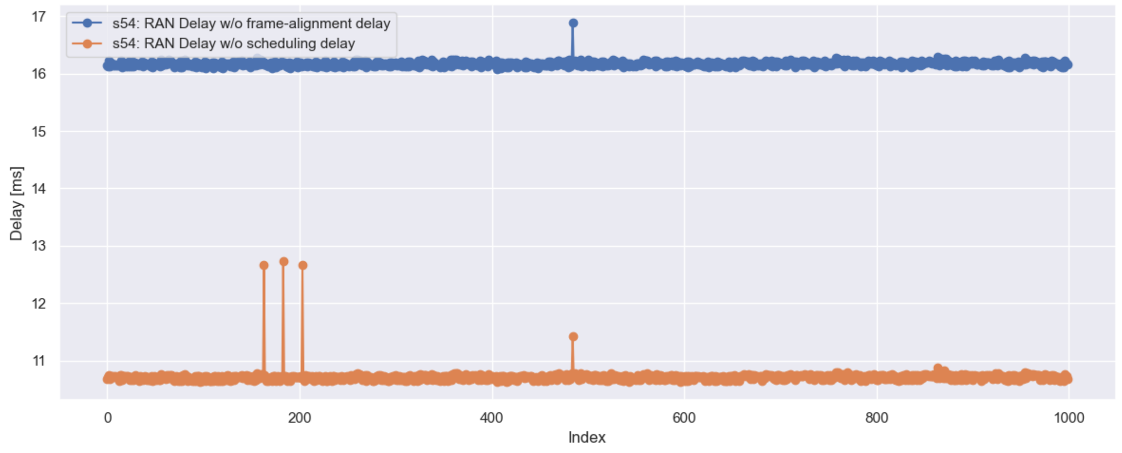


3.1.5 plotTimeSeries plot data values v.s. index

```
[16]: if IF_SHOW_USAGE == True:
          Meas_s54.plotTimeSeries(
              ["frame_alignment_delays", "scheduling_delays", "ran_delays"]
          )
```



```
[17]: if IF_SHOW_USAGE == True:
          Meas_s54.plotTimeSeries(
              ["RAN Delay w/o frame-alignment delay", "RAN Delay w/o scheduling␣
       ↪delay"],
              curves=[
                  Meas_s54.delays.ran_delays - Meas_s54.delays.frame_alignment_delays,
                  Meas_s54.delays.ran_delays - Meas_s54.delays.scheduling_delays,
              ],
              marker="o",
          )
```



1.4 4 class MultiMeas for visualizing multiple datasets

[18]:
```
class MultiMeas:
    def __init__(self, datasets_dir=DATASETS_DIR,␣
 ↪meas_labels=["s39","s40","s49"]):
        """
        MultiMeas:
            save and visualize multiple Meas objects

        Parameters:
            datasets_dir(str): Path of all datasets;
            meas_label(list of str): The measurement label;
        """
        self.meas=[]
        self.meas_labels=meas_labels
        for meas_label in meas_labels:
            self.meas.append(Meas(datasets_dir=datasets_dir,␣
 ↪meas_label=meas_label))
        self.folder_label="_".join(meas_labels)
        if not os.path.exists(PLOTS_DIR + self.folder_label):
            os.makedirs(PLOTS_DIR + self.folder_label)

    def plotCCDF(self, curve_names, meas_names, skip_first=SKIP_FIRST,␣
 ↪skip_last=SKIP_LAST, figsize=(8,5), plt_show=True, ax_external=None,␣
 ↪title='ccdf_plot'):
        """
        MultiMeas.plotCCDF:
            plot 1 ccdf among 13 kinds of delay measurement

        Parameters:
            curve_names(list of str):the delay measurement you want to plot
            meas_names(list of str):
            skip_first(int): skip first a few packets
            skip_last(int):  skip last a few packets
            figsize(tuple):  figsize
            plt_show(bool):  if or not show and save figure
            ax_external(object) ax object
            title(str): title (and filename)
        """
        if ax_external is None:
            _, ax=plt.subplots(figsize=figsize)
        else:
            ax=ax_external
        meas_list=[meas for meas in self.meas if meas.meas_label in meas_names]
        for one_meas in meas_list:
            for curve_name in curve_names:
                one_meas.plotCCDF(curve_name, skip_first=skip_first,␣
↪skip_last=skip_last,ax_external=ax, plt_show=False)
        if plt_show:
            i = 1
            while os.path.exists(os.path.join(PLOTS_DIR, self.folder_label,␣
↪f"{title}_{i}.png")):
                i += 1
            ax.set_title(f"{title}_{i}")
            plt.tight_layout()
            plt.savefig(
                f"{PLOTS_DIR}{self.folder_label}/{title}_{i}.png",
                dpi=300,
                bbox_inches="tight",
            )
            plt.show()

    def plotAllPerDelayType(self, figsize=(24,24),subplot_division=[1,1]):
        """
        Multieas.plotAllPerDelayType:
             plot CCDFs from multiple meas files on the same axes
        Parameters:
            figsize(tuple):  figsize
            subplot_division(list=[x,y]): x-row y-column subplots. If [1,1],␣
↪figures will be plotted separately.
        """
        curve_names = self.meas[0].listDelaysAttr()
        curve_names = [
            curve_name for curve_name in curve_names if curve_name !=␣
↪"timestamps"
        ]
        figure_num=0
        for i in range(0,len(curve_names)):
            if i % (subplot_division[0]*subplot_division[1]) == 0:
                if i != 0:
                    plt.tight_layout()
                    plt.savefig(f"{PLOTS_DIR}{self.folder_label}/
↪all_ccdf_plots_{figure_num}.png", dpi=300)
                    figure_num=figure_num+1
                    plt.show()
                _, axs = plt.subplots(subplot_division[0], subplot_division[1],␣
↪figsize=figsize)

            if subplot_division[0]*subplot_division[1]==1:
                self.plotCCDF([curve_names[i]],self.meas_labels,␣
↪plt_show=False, ax_external=axs)
            elif subplot_division[0]==1 or subplot_division[1]==1:
                self.plotCCDF([curve_names[i]],self.meas_labels,␣
↪plt_show=False, ax_external=axs[i %␣
↪(subplot_division[1]*subplot_division[0])])
            else:
                self.plotCCDF(
                    [curve_names[i]],
                    self.meas_labels,
                    plt_show=False,
                    ax_external=axs[
                        (
                            i
                            % (subplot_division[1] * subplot_division[0])
                            // subplot_division[1]
                        ),
                        i % subplot_division[1],
                    ],
                )
        plt.tight_layout()
        plt.savefig(f"{PLOTS_DIR}{self.folder_label}/
↪all_ccdf_plots_{figure_num}.png", dpi=300, bbox_inches="tight")
        plt.show()

    def plotHistograms(self, delay_name, ax_external=None,␣
↪skip_first=SKIP_FIRST, skip_last=SKIP_LAST, y_log=True, outlier=None,␣
↪figsize=(8,5)):
        """
        Plots histograms for multiple arrays side by side on the same axes with␣
↪normalized frequency.

        :param delay_name: The delay compoent to plot
        :param ax: Axes object to plot on
        :param skip_first: only take [skip_first:-skip_last] to plot
        :param skip_last:  only take [skip_first:-skip_last] to plot
        :param labels: List of labels for each array
        :param y_log: Boolean to set y-axis to log scale if True
        :param outlier: Cap value for outliers (optional)
        """

        values_per_meas = []
        labels = []
        for one_meas in self.meas:
            one_delay=getattr(one_meas.delays, delay_name, None)
            if one_delay is not None:
                values_per_meas.append(one_delay[skip_first:-skip_last])
                labels.append(one_meas.data.meas_label)
            else:
                print(f"No attribute {delay_name} found in meas.delays of␣
↪{one_meas.data.meas_label}.")

        if ax_external is not None:
            ax=ax_external
        else:
            _, ax=plt.subplots(figsize=figsize)

        plot_multiple_histograms(values_per_meas=values_per_meas, ax=ax,␣
↪labels=labels, y_log=y_log, delay_type_label=delay_name, outlier=outlier)
        plt.savefig(f"{PLOTS_DIR}{self.folder_label}/{delay_name}_hist_plot.
↪png", dpi=300, bbox_inches="tight")
        plt.show()

    def plotTimeSeries(self, curve_names, meas_names, start_idx=2500,␣
↪end_idx=2700, figsize=(12,5), marker="*", title="timeseries"):
        """
        Plot time series of multiple variables of multiple datasets in one plot.

        Parameters:
            curve_names(list of str): attribute names in Meas.Delays
            meas_names(list of str): dataset names
            start_idx(int): start index
            end_idx(int): end index
            figsize((int, int)): figsize
            marker(char): marker style of datapoints
        """
        _, ax = plt.subplots(figsize=figsize)

        meas_list=[meas for meas in self.meas if meas.meas_label in meas_names]
        for one_meas in meas_list:
            one_meas.plotTimeSeries(curve_names, start_idx=start_idx,␣
↪end_idx=end_idx, figsize=figsize, marker=marker, ax_external=ax,␣
↪plt_show=False)
        i = 0
        while os.path.exists(os.path.join(PLOTS_DIR, self.folder_label,␣
↪f"{title}_{i}.png")):
            i += 1
        plt.tight_layout()
        plt.savefig(
            f"{PLOTS_DIR}{self.folder_label}/{title}_{i}.png", dpi=300,␣
↪bbox_inches="tight"
        )
        plt.show()
    def dataFrame(self, curve_names, curve_labels=None, csv_path=None,␣
↪if_save=True):
        """
        Export dataframe of given delay components with given labels, and save␣
↪it to csv file.

        curve_labels(list of str): data labels
        curve_names(list of str): attributes names of Meas.Delay (use Meas.
↪listDelaysAttr() to check)
        csv_path(str): Optional. default: the data folder of the meas.
        """

        for dataset in self.meas:
            data = dict()

            for idx, curve_name in enumerate(curve_names):
                curve = getattr(dataset.delays, curve_name, None)
                if curve is not None:
                    if curve_labels is not None:
                        data[curve_labels[idx]] = curve
                    else:
                        data[curve_names[idx]] = curve

            # Create a pandas DataFrame with each array as a column
            df = pd.DataFrame(data)
            # Display the first few rows of the DataFrame
            print(df.head())

            # Export the DataFrame to a CSV file
            if csv_path == None:
                csv_path = f"{DATASETS_DIR}csv/"
                print(csv_path)

            if if_save == True:
                df.to_csv(
                    os.path.join(csv_path, f"{dataset.data.meas_label}_{"_".
↪join(curve_labels)}.csv"), index=True
                )
                print(
                    f"Dataframe saved to {os.path.join(csv_path, f"{dataset.
↪data.meas_label}_{"_".join(curve_labels)}.csv")}"
                )

        return
```

1.4.1 4.1 Usage of class MultiMeas

4.1.1 Import datasets

[19]:
```
if IF_SHOW_USAGE == True:
    Meas_s39_s40_s54 = MultiMeas(meas_labels=["s39", "s40", "s54"])
```

```
RNTIs in packets of s39: ['dc46']
For packet with 21767 in s39, tbs is None but segments is not None, Remove this
packet!
2024-12-19 15:55:16.802 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:55:16.969 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
2024-12-19 15:55:17.324 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:55:17.470 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
2024-12-19 15:55:17.791 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:55:17.941 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
2024-12-19 15:55:19.345 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:55:19.455 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
2024-12-19 15:55:19.903 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:55:20.171 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
2024-12-19 15:55:20.752 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
```

```
decomp:get_tx_delay:246 - Packet 1940

phy.in_t or phy.in_t not present
2024-12-19 15:56:04.833 | ERROR    |
decomp:get_tx_delay:246 - Packet 1098

phy.in_t or phy.in_t not present
```

[20]:
```
if IF_SHOW_USAGE == True:
    print(Meas_s39_s40_s54.meas[0].data.meas_label)
    print(Meas_s39_s40_s54.meas_labels)
```

```
s39
['s39', 's40', 's54']
```

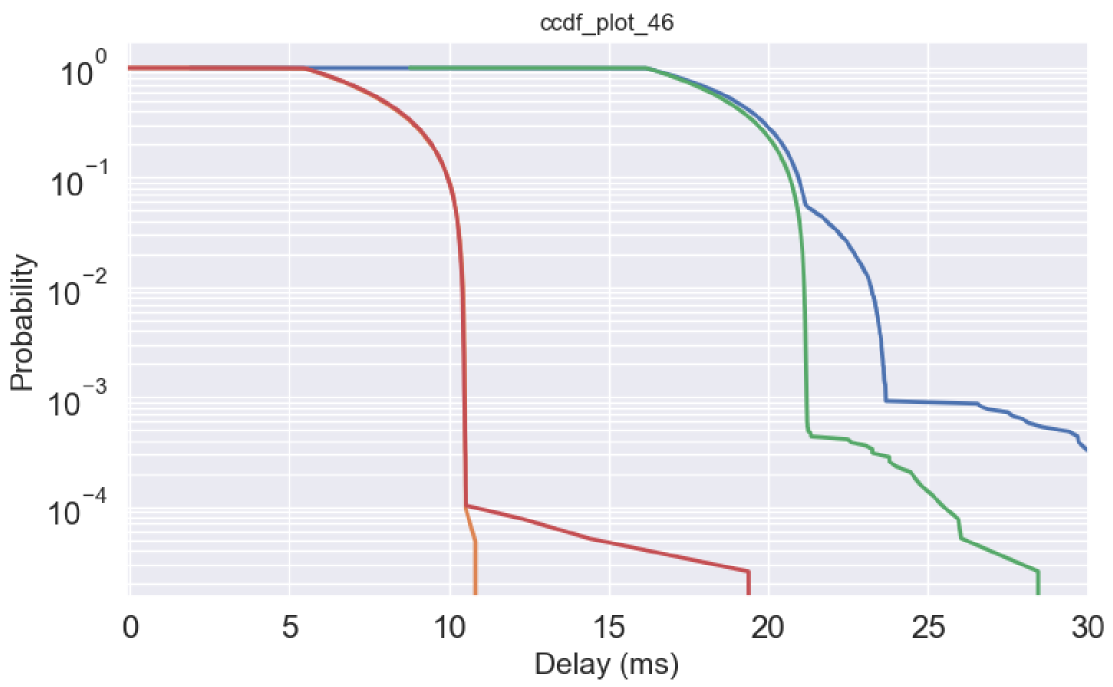
4.1.2 check Data

[21]:
```
if IF_SHOW_USAGE == True:
    Meas_s39_s40_s54.meas[0].checkData("packets")
```

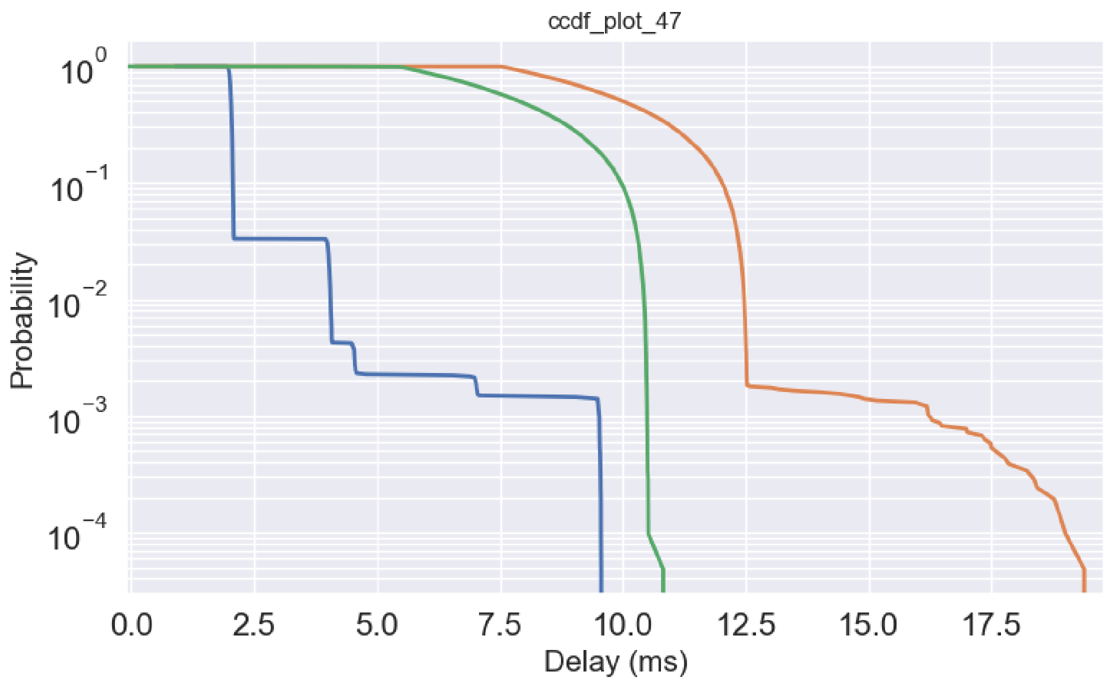
```
{
    "sn": 1,
    "id": 21766,
    "len": 48,
    "ip.in_t": 1729237468.921818,
    "ip.out_t": 1729237468.927717,
    "rlc.in_t": 1729237468.921827,
    "rlc.out_t": 1729237468.927699,
    "backlog": 0,
    "rlc.attempts": [
        {
            "id": 0,
            "so": 0,
            "len": 51,
            "rep_acked": true,
            "resegment": [
                NaN,
                NaN,
                NaN,
                NaN
            ],
            "repeated": false,
            "mac.in_t": 1729237468.926534,
            "mac.out_t": 1729237468.927699,
            "rnti": "dc46",
            "frame": 293,
            "slot": 4,
            "acked": true,
            "mac.attempts": [
                {
                    "len": 116,
                    "id": 75,
                    "rnti": "dc46",
                    "frame": 293,
                    "slot": 4,
                    "hqpid": 10,
                    "phy.in_t": 1729237468.92654,
                    "phy.out_t": 1729237468.927645,
                    "acked": true,
                    "hqround": 0,
                    "next_id": null,
                    "prev_id": null
                }
            ]
        }
    ]
}
```

4.1.3 `plotCCDF` plot CCDFs of certain delay components from certain datasets

[22]:
```
if IF_SHOW_USAGE == True:
    Meas_s39_s40_s54.plotCCDF(
        ["ran_delays", "scheduling_delays"], ["s39", "s49", "s54"]
    )
```

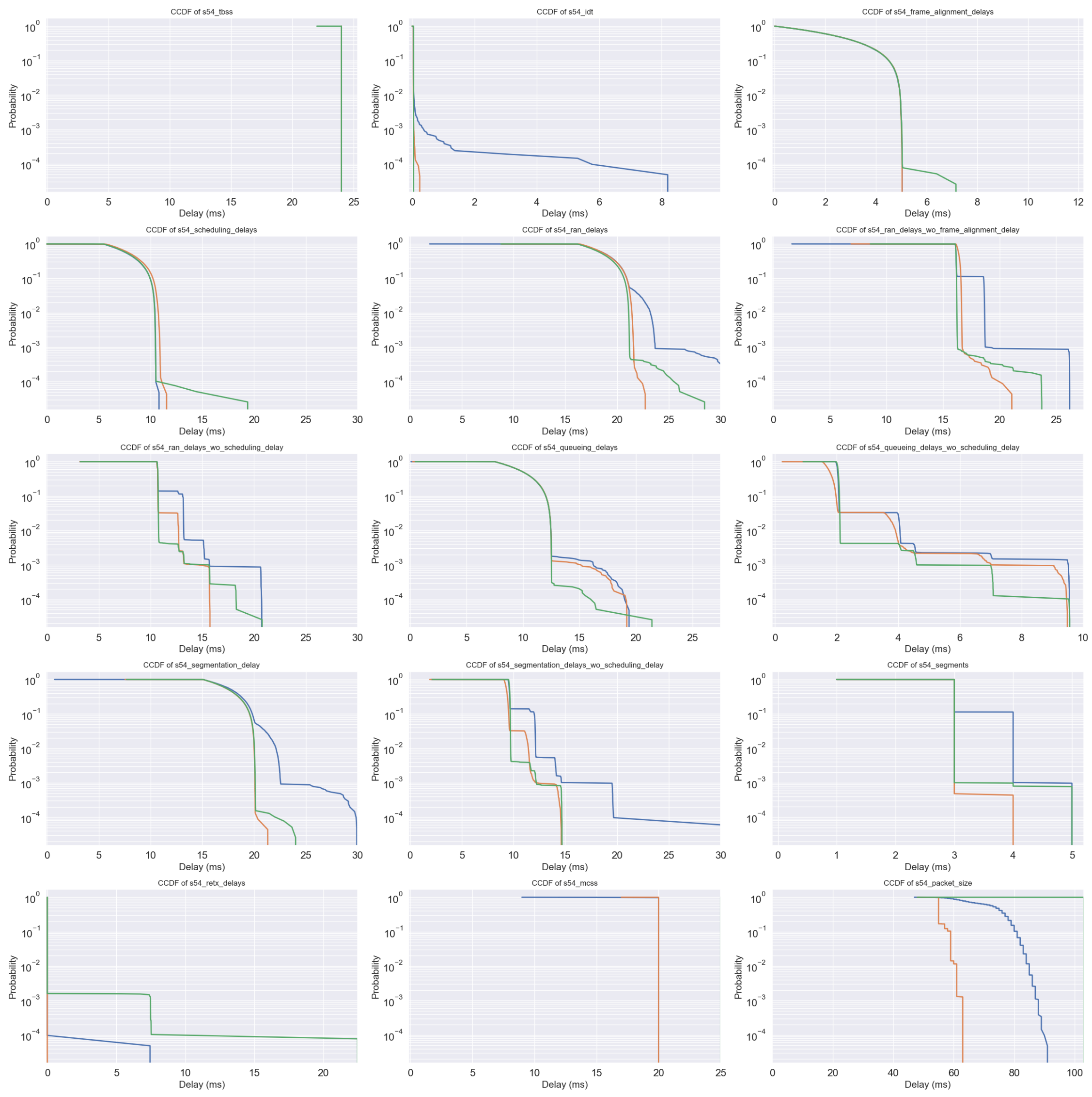


[23]:
```
if IF_SHOW_USAGE == True:
    Meas_s39_s40_s54.plotCCDF(
        ["queueing_delays_wo_scheduling_delay", "queueing_delays",␣
 ↪"scheduling_delays"],
        ["s39"],
    )
```



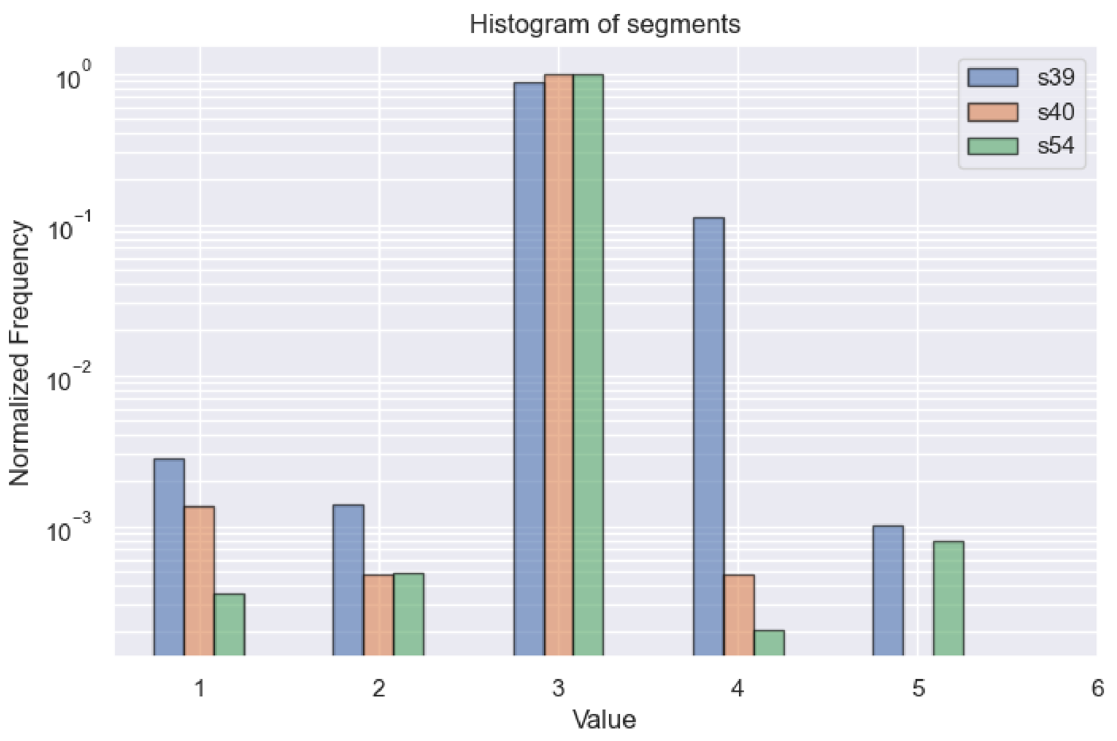
4.1.4 `plotAllPerDelayType` plot CCDFs from multiple meas files on the same axes

[24]:
```
if IF_SHOW_USAGE == True:
    Meas_s39_s40_s54.plotAllPerDelayType(subplot_division=[5, 3])
```

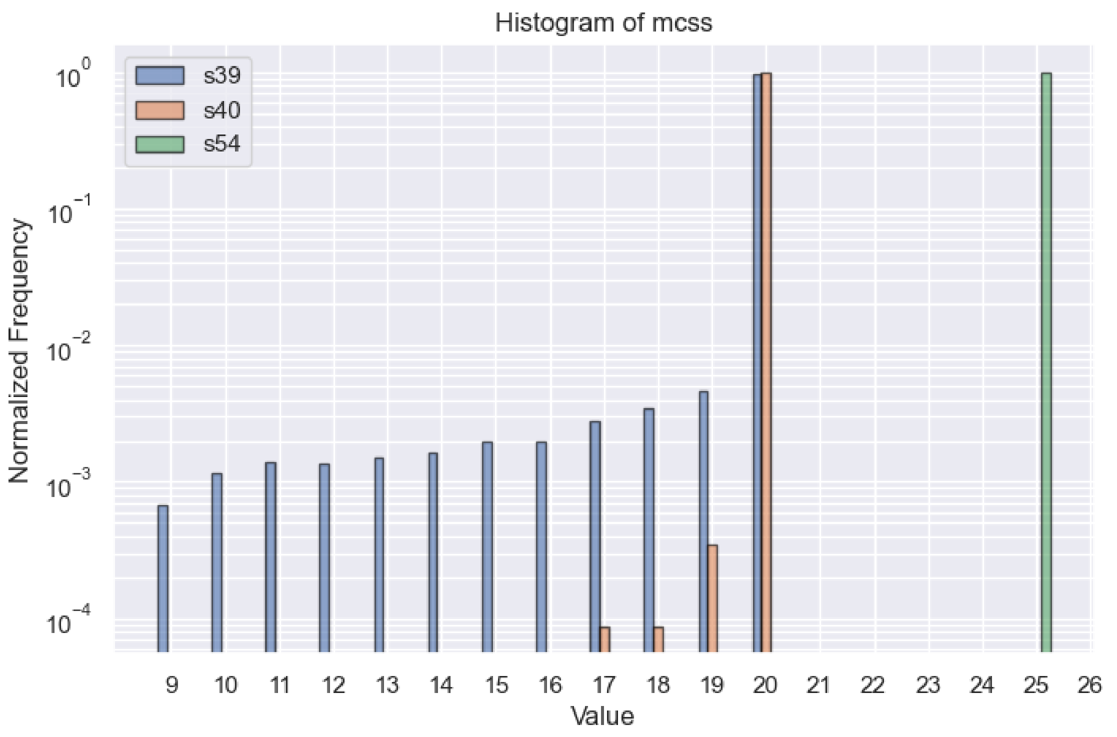


4.1.5 `plotHistograms` Plots histograms for multiple arrays side by side on the same axes with normalized frequency.

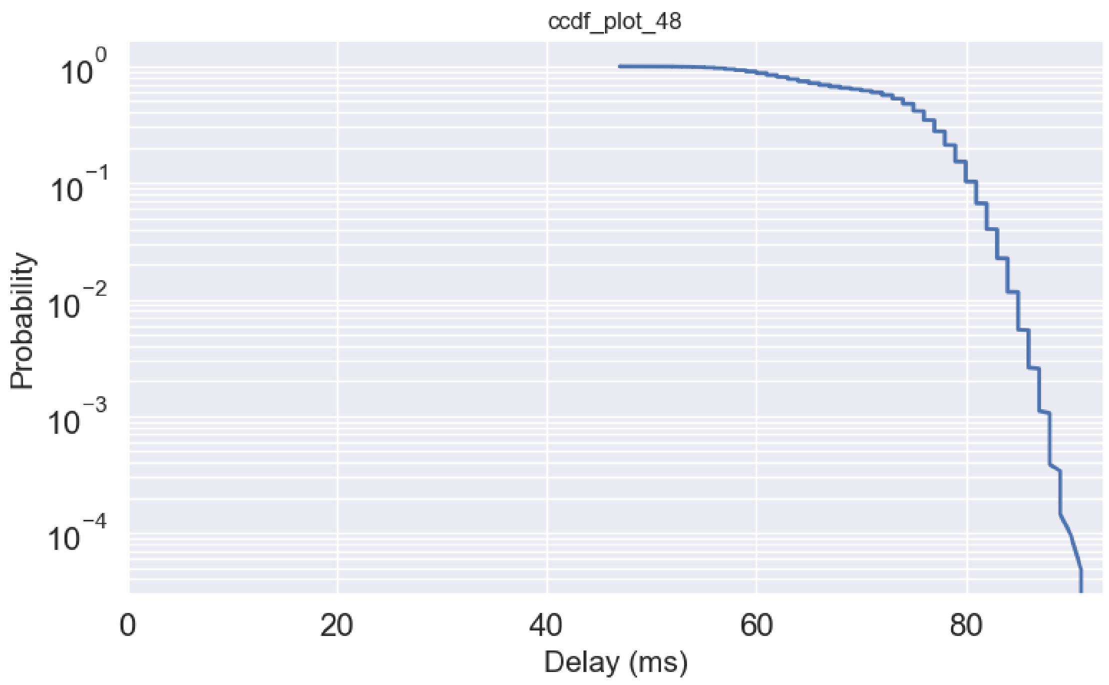
[25]:
```
if IF_SHOW_USAGE == True:
    Meas_s39_s40_s54.plotHistograms("segments")
```



[26]:
```
if IF_SHOW_USAGE == True:
    Meas_s39_s40_s54.plotHistograms("mcss")
```

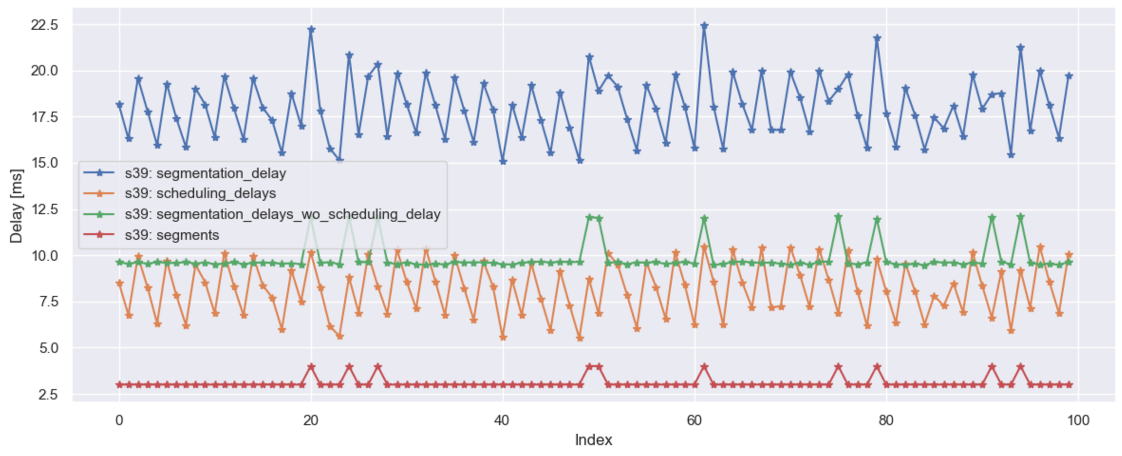


```
[27]: if IF_SHOW_USAGE == True:
          Meas_s39_s40_s54.plotCCDF(["packet_size"], ["s39"])
```

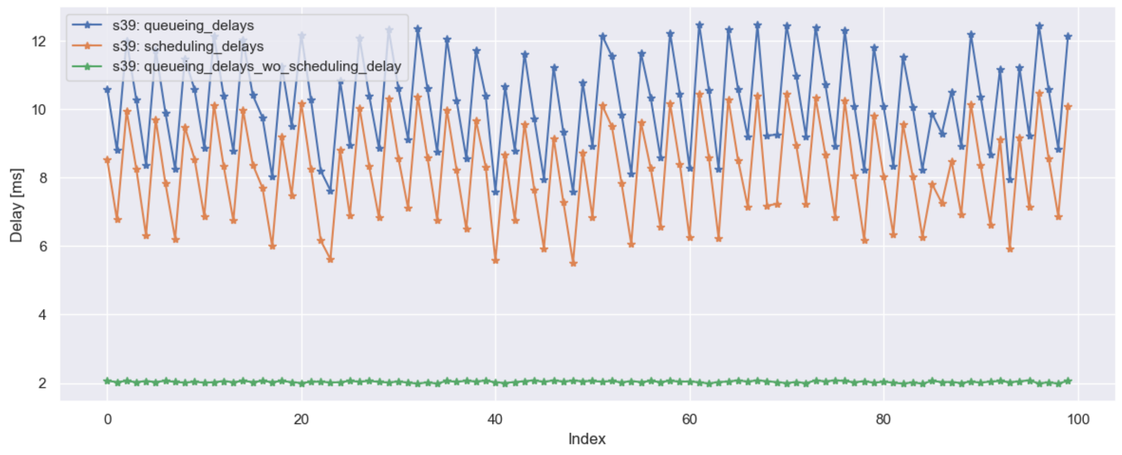


4.1.6 plotTimeSeries

```
[28]: if IF_SHOW_USAGE == True:
          Meas_s39_s40_s54.plotTimeSeries(
              [
                  "segmentation_delay",
                  "scheduling_delays",
                  "segmentation_delays_wo_scheduling_delay",
                  "segments",
              ],
              ["s39"],
              start_idx=2500,
              end_idx=2600,
          )
```



```
[29]: if IF_SHOW_USAGE == True:
          Meas_s39_s40_s54.plotTimeSeries(
              [
                  "queueing_delays",
                  "scheduling_delays",
                  "queueing_delays_wo_scheduling_delay",
              ],
              ["s39"],
              start_idx=2500,
              end_idx=2600,
          )
```



1.5 5 Export to DataFrame

[30]:
```
if IF_SHOW_USAGE == True:
    df = Meas_s39_s40_s54.meas[0].dataFrame(
        [
            "tbss",
            "segments",
            "packet_size",
            "timestamps",
            "segmentation_delays_wo_scheduling_delay",
            "queueing_delays_wo_scheduling_delay",
        ],  # attrbutes name of Meas.delays
        [
            "TBS",
            "SegmentsNum",
            "PacketSize",
            "TimeStamps",
            "SegmentDelay(noSched)",
            "QueueingDelay(noSched)",
        ],  # labels to display (optional)
    )
```

```
   TBS  SegmentsNum  PacketSize  TimeStamps  SegmentDelay(noSched)  \
0  116            1          48       21766               2.076864
1  116            1          48       21765               2.111197
2  116            1          60       21764               2.075911
3   24            2          52       21763               9.627104
4  145            1          53       21762               2.051115

   QueueingDelay(noSched)
0                2.025843
1                2.021074
2                2.002001
3                2.027035
4                1.973152
./data/s39
Dataframe saved to ./data/s39\TBS_SegmentsNum_PacketSize_TimeStamps_SegmentDelay
(noSched)_QueueingDelay(noSched).csv
```

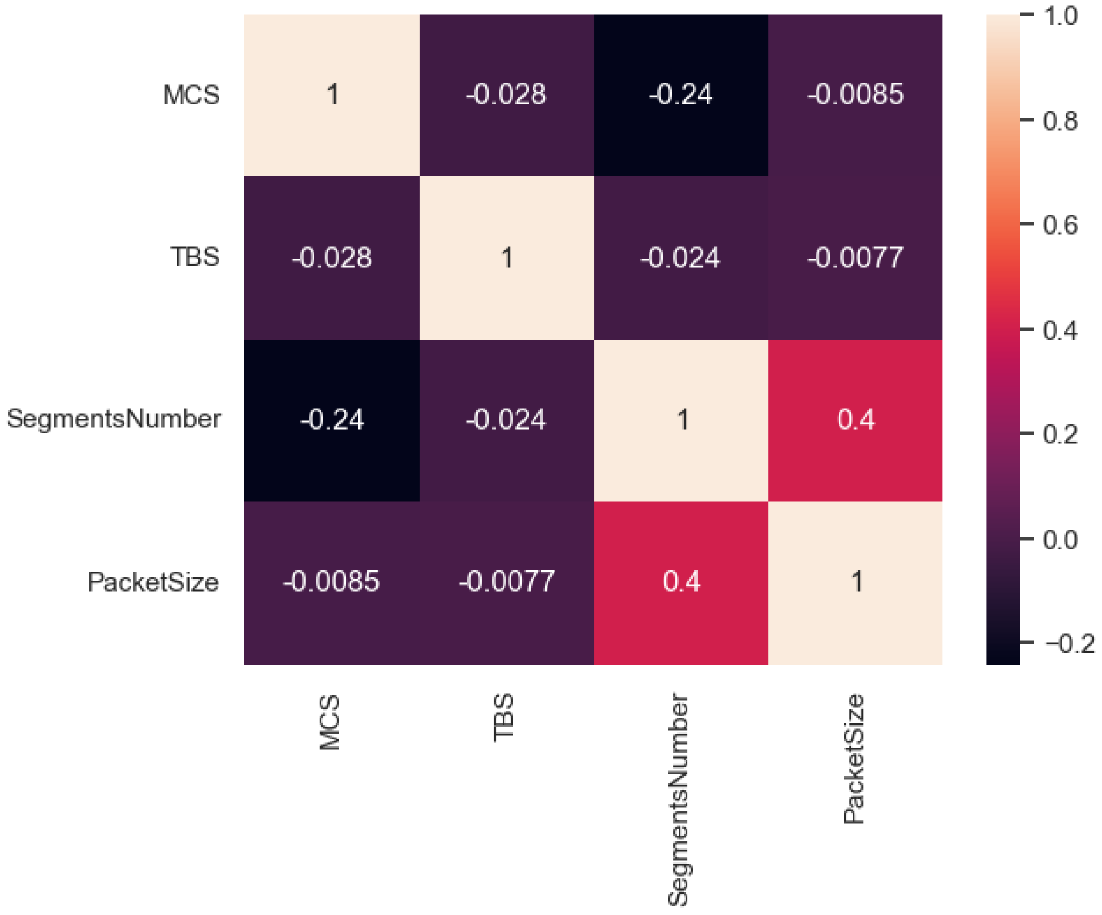
1.6 6 Show Correlation

1.6.1 6.1 plotCrossCorrelation Correlation Coef Matrix

[31]:
```
if IF_SHOW_USAGE == True:
    Meas_s39_s40_s54.meas[0].plotCrossCorrelation(
        ["mcss", "tbss", "segments", "packet_size"],  # attrbutes name of Meas.
 ↪delays
```

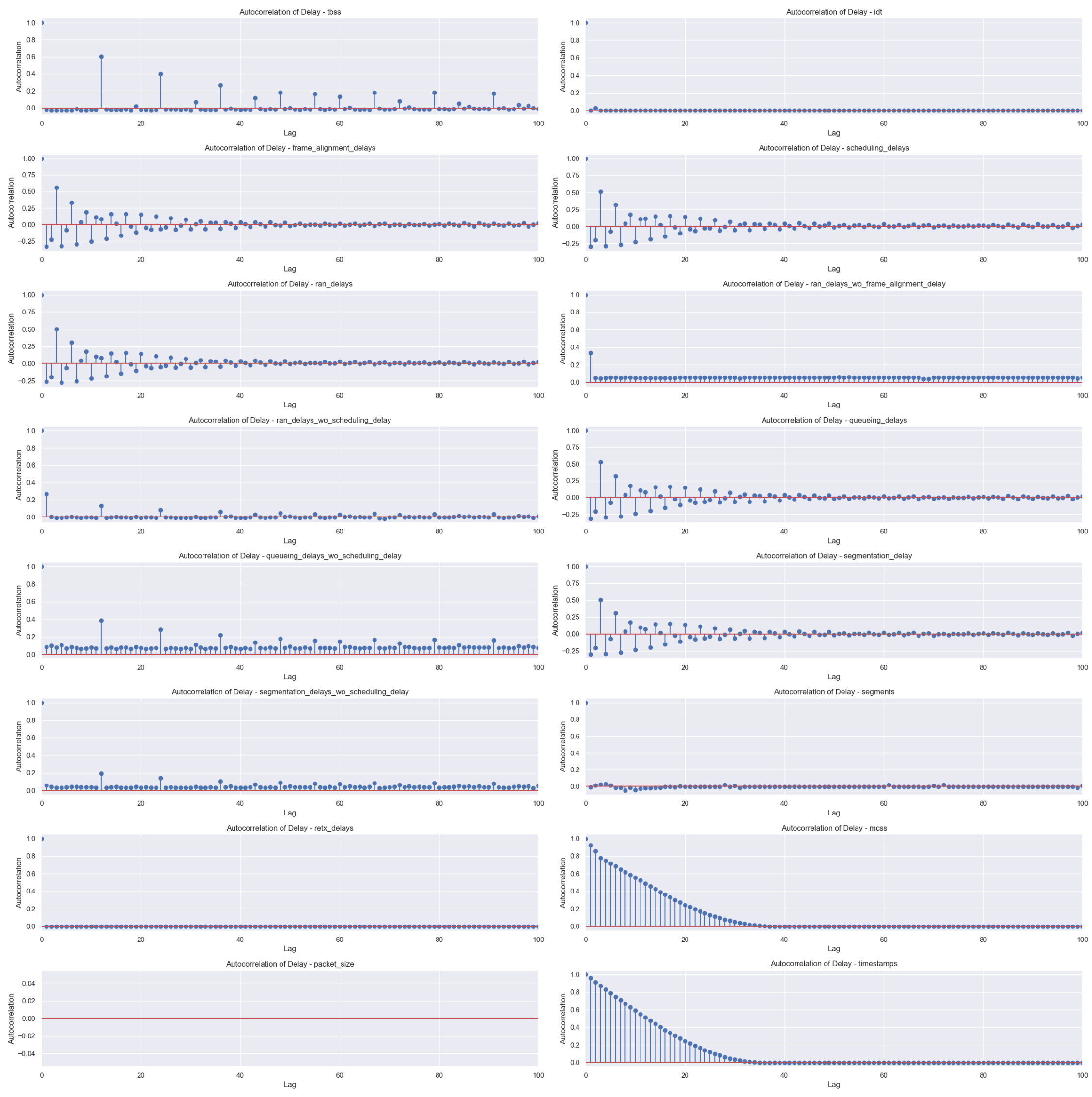
```
        ["MCS", "TBS", "SegmentsNumber", "PacketSize"],  # labels to display␣
↪(optional)
    )
```

```
   MCS  TBS  SegmentsNumber  PacketSize
0    9  116               1          48
1    9  116               1          48
2    9  116               1          60
3    9   24               2          52
4    9  145               1          53
./data/s39
```



```
[32]: if IF_SHOW_USAGE == True:
          Meas_s39_s40_s54.meas[1].plotAllAutoCorrelation(

              subplot_division=[8, 2],

          )
```

```
c:\Users\18263\.conda\envs\ProjectCourse_5GDelay\Lib\site-
packages\statsmodels\tsa\stattools.py:693: RuntimeWarning: invalid value
encountered in divide
  acf = avf[: nlags + 1] / avf[0]
```

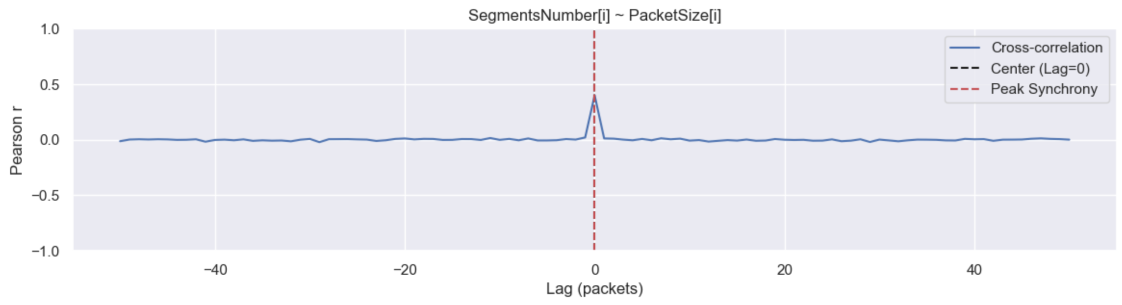


1.6.2 6.2 plotTLCC(d1, d2)

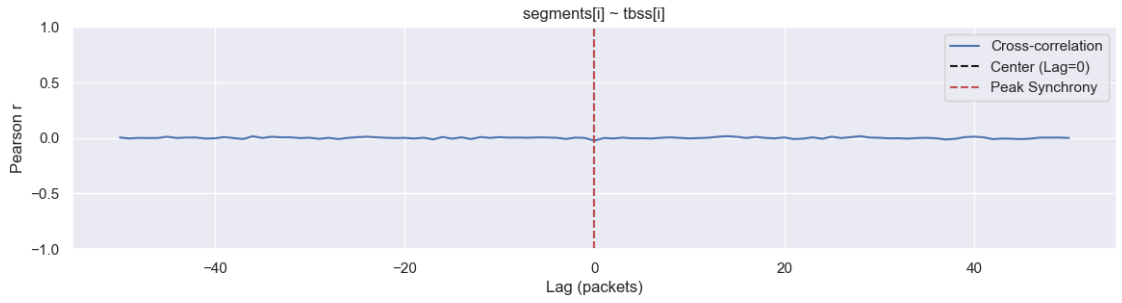
offset =-2 ⇒ `d1[i]` impacted by `d2[i-2]`

offset = 1 ⇒ `d1[i]` impacted by `d2[i+1]` (wrong) ⇒ `d2[i]` impacted by `d1[i-1]` (true)

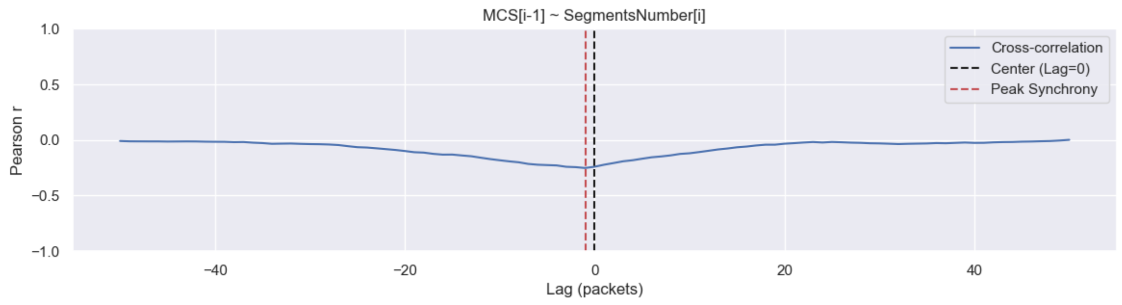
[33]:
```
if IF_SHOW_USAGE == True:
    Meas_s39_s40_s54.meas[0].plotTLCC(
        ["segments", "packet_size"],  # attrbutes name of Meas.delays
        ["SegmentsNumber", "PacketSize"],  # labels to display (optional)
    )
```



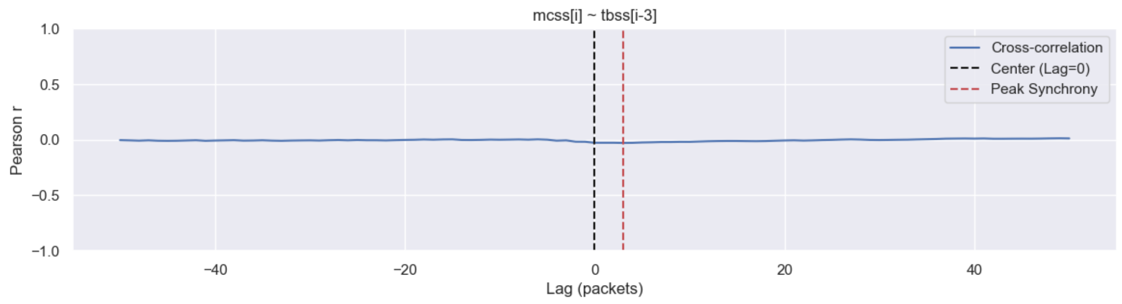
[34]:
```
if IF_SHOW_USAGE == True:
    Meas_s39_s40_s54.meas[0].plotTLCC(

        ["segments", "tbss"],  # attrbutes name of Meas.delays

    )

# number of segments[i] ~ TBS [i-2]
```



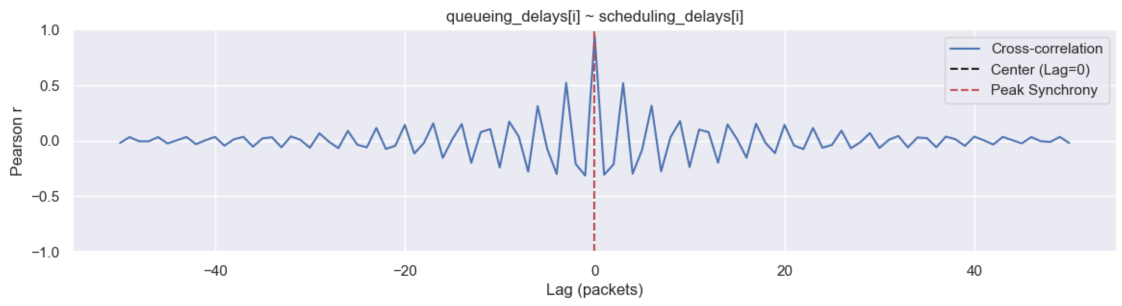
[35]:
```
if IF_SHOW_USAGE == True:
    Meas_s39_s40_s54.meas[0].plotTLCC(
        ["mcss", "segments"],  # attrbutes name of Meas.delays
        ["MCS", "SegmentsNumber"],  # labels to display (optional)
    )
```



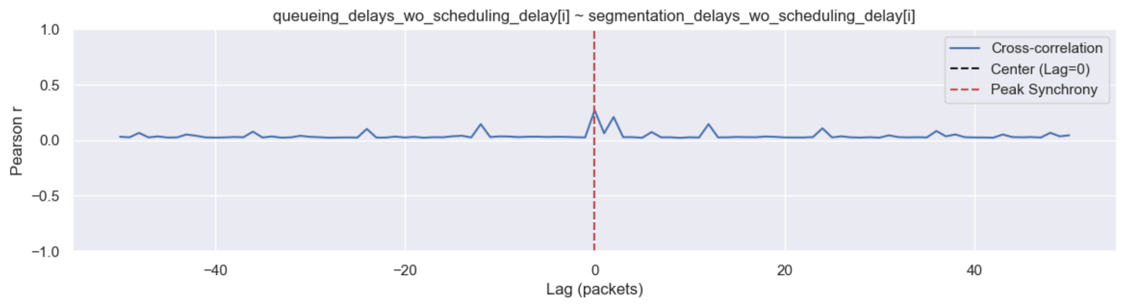
```
[36]: if IF_SHOW_USAGE == True:
          Meas_s39_s40_s54.meas[0].plotTLCC(

              ["mcss", "tbss"],  # attrbutes name of Meas.delays

          )
```



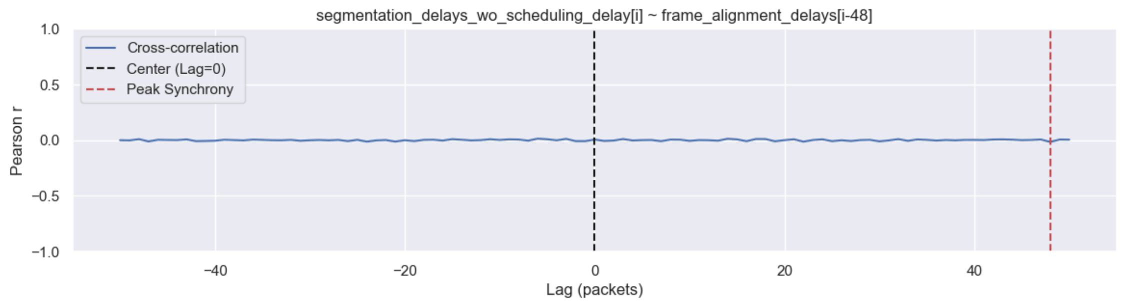
```
[37]: if IF_SHOW_USAGE == True:
          Meas_s39_s40_s54.meas[1].plotTLCC(

              ["queueing_delays", "scheduling_delays"]  # attrbutes name of Meas.
      ↪delays

          )
```



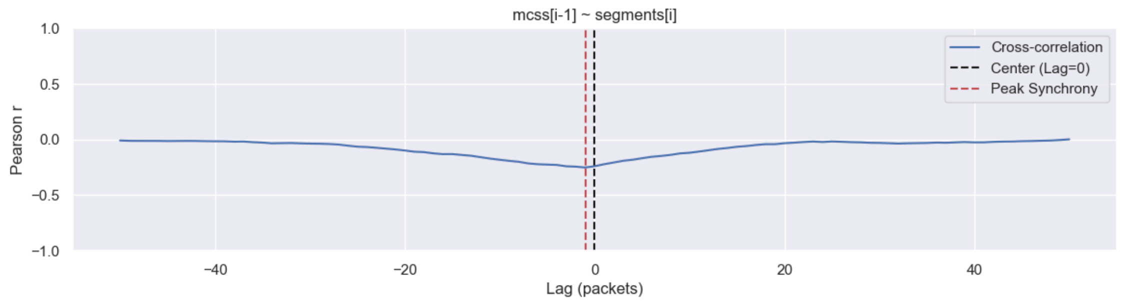
```
[38]: if IF_SHOW_USAGE == True:
          Meas_s39_s40_s54.meas[1].plotTLCC(
              [
                  "queueing_delays_wo_scheduling_delay",
                  "segmentation_delays_wo_scheduling_delay",
              ],  # attrbutes name of Meas.delays
          )
```



```
[39]: if IF_SHOW_USAGE == True:
          Meas_s39_s40_s54.meas[1].plotTLCC(
              [
                  "segmentation_delays_wo_scheduling_delay",
                  "frame_alignment_delays",
              ],  # attrbutes name of Meas.delays
          )
```



```
[40]: if IF_SHOW_USAGE == True:
          Meas_s39_s40_s54.meas[0].plotTLCC(

              [
                  "mcss",

                  "segments",

              ],  # attrbutes name of Meas.delays

          )
```



1.6.3 Test

[41]:
```
if IF_SHOW_USAGE == True:
    Meas_s39_s40_s54.meas[0].listDataAttr()
```

1.6.4 Show length

[42]:
```
if IF_SHOW_USAGE == True:
    print(list(vars(Meas_s39_s40_s54.meas[0].delays).keys()))  # ['attr1',␣
 ↪'attr2']

    for attr_name in list(vars(Meas_s39_s40_s54.meas[0].delays).keys()):
        attr = getattr(Meas_s39_s40_s54.meas[0].delays, attr_name, None)
        if attr is not None:
            print(
                f"len(Meas_{Meas_s39_s40_s54.meas[0].data.meas_label}.delays.
 ↪{attr_name})= {len(attr)} "
            )
# print(len(Meas_s39_s40_s49.meas[1].delays.queueing_delays))
# print(len(Meas_s39_s40_s49.meas[1].delays.frame_alignment_delays))
# print(len(Meas_s39_s40_s49.meas[1].delays.segmentation_delay))
# print(len(Meas_s39_s40_s49.meas[1].delays.mcss))
```

```
['tbss', 'idt', 'frame_alignment_delays', 'scheduling_delays', 'ran_delays',
'ran_delays_wo_frame_alignment_delay', 'ran_delays_wo_scheduling_delay',
'queueing_delays', 'queueing_delays_wo_scheduling_delay', 'segmentation_delay',
'segmentation_delays_wo_scheduling_delay', 'segments', 'retx_delays', 'mcss',
'packet_size', 'timestamps']
```

```
len(Meas_s39.delays.tbss)= 21767
len(Meas_s39.delays.idt)= 21766
len(Meas_s39.delays.frame_alignment_delays)= 21767
len(Meas_s39.delays.scheduling_delays)= 21767
len(Meas_s39.delays.ran_delays)= 21767
len(Meas_s39.delays.ran_delays_wo_frame_alignment_delay)= 21767
len(Meas_s39.delays.ran_delays_wo_scheduling_delay)= 21767
len(Meas_s39.delays.queueing_delays)= 21767
len(Meas_s39.delays.queueing_delays_wo_scheduling_delay)= 21767
len(Meas_s39.delays.segmentation_delay)= 21767
len(Meas_s39.delays.segmentation_delays_wo_scheduling_delay)= 21767
len(Meas_s39.delays.segments)= 21767
len(Meas_s39.delays.retx_delays)= 21767
len(Meas_s39.delays.mcss)= 21767
len(Meas_s39.delays.packet_size)= 21767
len(Meas_s39.delays.timestamps)= 21767
```

[43]:
```
if IF_SHOW_USAGE == True:
    queueing_delay   = Meas_s39_s40_s54.meas[0].delays.queueing_delays
    scheduling_delay = Meas_s39_s40_s54.meas[0].delays.scheduling_delays

    diff = queueing_delay - scheduling_delay
    diff = diff[diff<0]
    len(diff)
    print(diff)
```

```
[]
```

[44]:
```
if IF_SHOW_USAGE == True:
    len(Meas_s39_s40_s54.meas[0].data.packets)
```

[45]:
```
if IF_SHOW_USAGE == True:
    Meas_s39 = Meas(meas_label="s39")
```

```
RNTIs in packets of s39: ['dc46']
For packet with 21767 in s39, tbs is None but segments is not None, Remove this
packet!
2024-12-19 15:57:01.709 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:57:01.816 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
2024-12-19 15:57:02.115 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:57:02.237 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
2024-12-19 15:57:02.601 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:57:02.728 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
2024-12-19 15:57:03.976 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:57:04.157 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
2024-12-19 15:57:04.486 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:57:04.653 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
2024-12-19 15:57:05.167 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:57:05.168 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:57:05.418 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
2024-12-19 15:57:05.419 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
2024-12-19 15:57:06.061 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:57:06.062 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:57:06.358 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
2024-12-19 15:57:06.360 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
2024-12-19 15:57:06.921 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:57:06.979 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
2024-12-19 15:57:07.582 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:57:07.683 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
2024-12-19 15:57:07.912 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:57:07.966 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
2024-12-19 15:57:08.135 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:57:08.200 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
```

[46]:
```
if IF_SHOW_USAGE == True:
    print(Meas_s39.data.sched_sched_sorted_dict[1729237464.116453])
```

```
{'decision_ts': 1729237464.113453, 'schedule_ts': 1729237464.116453,
'symbols_start': 10, 'symbols_num': 3, 'prbs_start': 0, 'prbs_num': 5, 'cause':
{'rnti': 'dc46', 'tbs': 24, 'mcs': 9, 'rbs': 5, 'type': 3, 'diff': 16724.0,
'buf': nan, 'sched': nan, 'hqround': nan, 'hqpid': nan}}
```

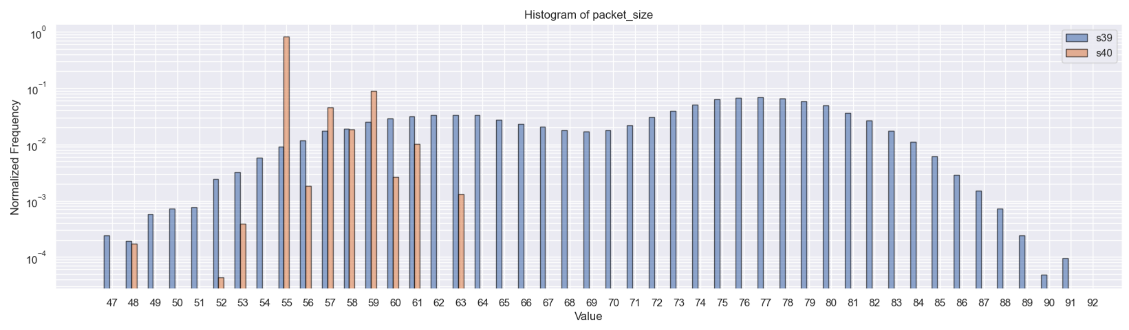
[47]:
```
if IF_SHOW_USAGE == True:
    Meas_s39_s40 = MultiMeas(meas_labels=["s39", "s40"])
```

```
RNTIs in packets of s39: ['dc46']
For packet with 21767 in s39, tbs is None but segments is not None, Remove this
```

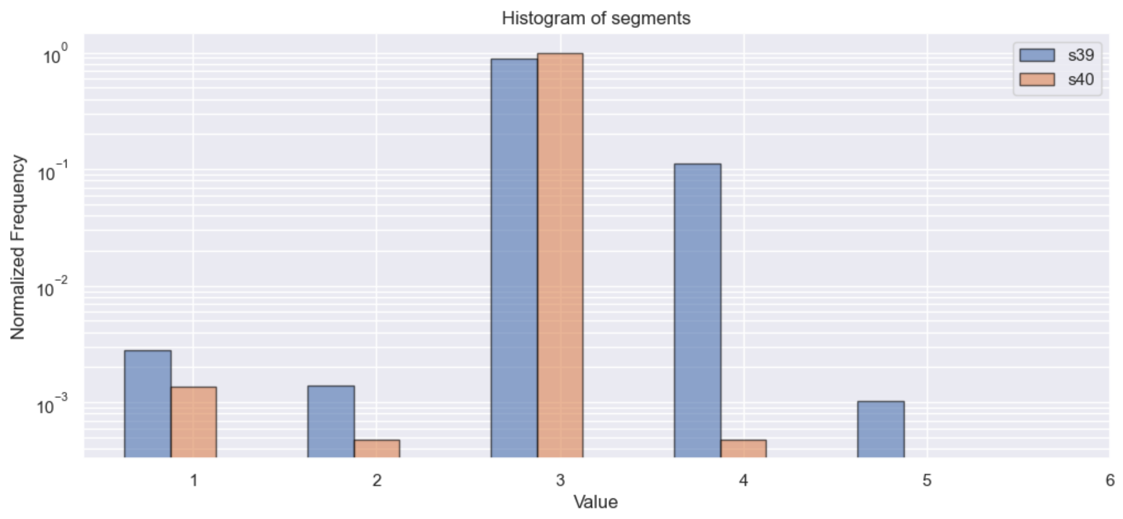
```
packet!
2024-12-19 15:57:13.051 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:57:13.150 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
2024-12-19 15:57:13.446 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:57:13.570 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
2024-12-19 15:57:13.884 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:57:14.034 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
2024-12-19 15:57:15.528 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:57:15.614 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
2024-12-19 15:57:15.881 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:57:16.047 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
2024-12-19 15:57:16.517 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:57:16.518 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138

phy.in_t or phy.in_t not present
2024-12-19 15:57:16.744 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013

phy.in_t or phy.in_t not present
```

```
2024-12-19 15:57:16.745 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013
phy.in_t or phy.in_t not present
2024-12-19 15:57:17.409 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138
phy.in_t or phy.in_t not present
2024-12-19 15:57:17.410 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138
phy.in_t or phy.in_t not present
2024-12-19 15:57:17.740 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013
phy.in_t or phy.in_t not present
2024-12-19 15:57:17.743 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013
phy.in_t or phy.in_t not present
2024-12-19 15:57:18.279 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138
phy.in_t or phy.in_t not present
2024-12-19 15:57:18.333 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013
phy.in_t or phy.in_t not present
2024-12-19 15:57:18.922 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138
phy.in_t or phy.in_t not present
2024-12-19 15:57:19.020 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013
phy.in_t or phy.in_t not present
2024-12-19 15:57:19.219 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138
phy.in_t or phy.in_t not present
2024-12-19 15:57:19.270 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013
phy.in_t or phy.in_t not present
2024-12-19 15:57:19.402 | ERROR    |
decomp:get_tx_delay:246 - Packet 16138
phy.in_t or phy.in_t not present
2024-12-19 15:57:19.460 | ERROR    |
decomp:get_tx_delay:246 - Packet 10013
phy.in_t or phy.in_t not present
RNTIs in packets of s40: ['9afe']
```

[48]:
```
if IF_SHOW_USAGE == True:
    Meas_s39_s40.plotHistograms("packet_size", figsize=[20, 5])
```



[49]:
```
if IF_SHOW_USAGE == True:
    Meas_s39_s40.plotHistograms("segments", figsize=[12, 5])
```



2 7 s39, s40, s59~s66

[50]:
```
Meas_s39_40_s59s66 = MultiMeas(meas_labels=["s39","s40","s59",␣
↪"s61","s62","s63","s64","s65","s66"])
```

```
RNTIs in packets of s39: ['dc46']
For packet with 21767 in s39, tbs is None but segments is not None, Remove this
packet!
2024-12-19 15:57:46.097 | ERROR    |
decomp:get_tx_delay:246 - Packet 3729

phy.in_t or phy.in_t not present
2024-12-19 16:00:48.971 | ERROR    |
decomp:get_tx_delay:246 - Packet 142

phy.in_t or phy.in_t not present
2024-12-19 16:00:48.975 | ERROR    |
decomp:get_tx_delay:246 - Packet 142

phy.in_t or phy.in_t not present
```

[51]:
```
item = Meas_s39_40_s59s66.meas[0]
for attr_name in list(vars(item.delays).keys()):
    attr = getattr(item.delays, attr_name, None)
    if attr is not None:
        print(
            f"len(Meas_{item.data.meas_label}.delays.{attr_name})= {len(attr)} "
        )
```

```
len(Meas_s39.delays.tbss)= 21767
len(Meas_s39.delays.idt)= 21766
len(Meas_s39.delays.frame_alignment_delays)= 21767
len(Meas_s39.delays.scheduling_delays)= 21767
len(Meas_s39.delays.ran_delays)= 21767
len(Meas_s39.delays.ran_delays_wo_frame_alignment_delay)= 21767
len(Meas_s39.delays.ran_delays_wo_scheduling_delay)= 21767
len(Meas_s39.delays.queueing_delays)= 21767
len(Meas_s39.delays.queueing_delays_wo_scheduling_delay)= 21767
len(Meas_s39.delays.segmentation_delay)= 21767
len(Meas_s39.delays.segmentation_delays_wo_scheduling_delay)= 21767
len(Meas_s39.delays.segments)= 21767
len(Meas_s39.delays.retx_delays)= 21767
len(Meas_s39.delays.mcss)= 21767
len(Meas_s39.delays.packet_size)= 21767
len(Meas_s39.delays.timestamps)= 21767
```

[52]:
```
diff = Meas_s39_40_s59s66.meas[1].delays.segmentation_delays_wo_scheduling_delay

neg_num=len([item for item in diff if item < 0 ])
print(f"{Meas_s39_40_s59s66.meas[1].meas_label}:{neg_num} negative values ")
```

```
s40:0 negative values
```

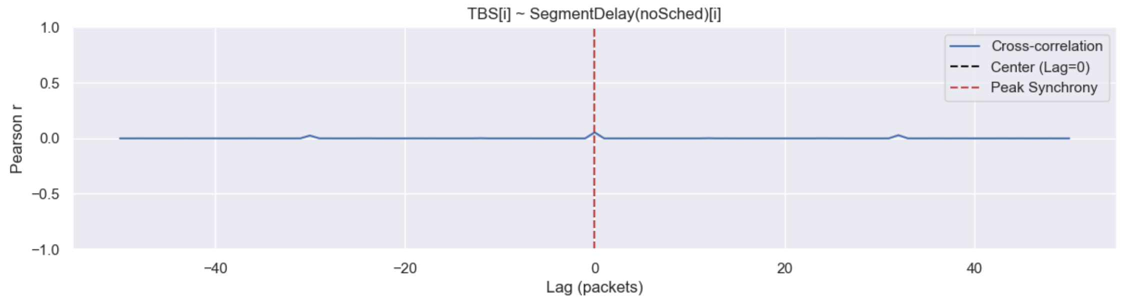
[53]:
```
df = Meas_s39_40_s59s66.meas[0].dataFrame(
    [
        "tbss",
        "segments",
        "packet_size",
        "segmentation_delays_wo_scheduling_delay"
    ],  # attrbutes name of Meas.delays
    [
        "TBS",
        "SegmentsNum",
        "PacketSize",
        "SegmentDelay(noSched)",
    ],  # labels to display (optional)
)
```

```
   TBS  SegmentsNum  PacketSize  SegmentDelay(noSched)
0  116            1          48               2.076864
1  116            1          48               2.111197
2  116            1          60               2.075911
3   24            2          52               9.627104
4  145            1          53               2.051115
./data/s39
Dataframe saved to
./data/s39\TBS_SegmentsNum_PacketSize_SegmentDelay(noSched).csv
```

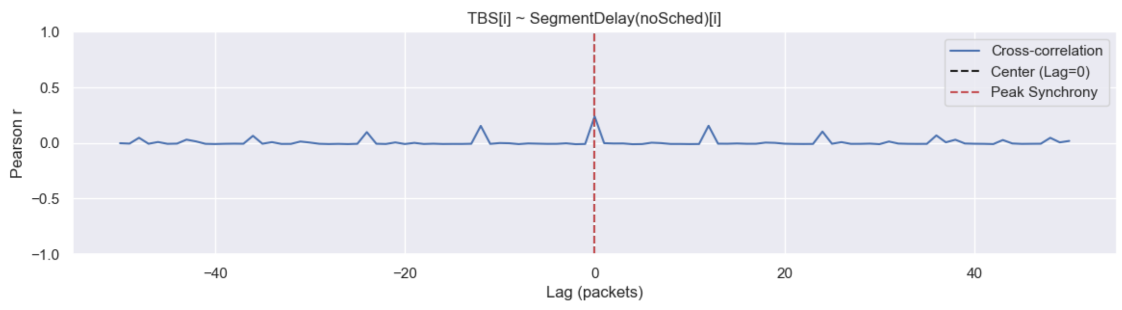
TBS vs SegmentDelay

```
[ ]: for i in range(0, len(Meas_s39_40_s59s66.meas_labels)):
         print(f"{Meas_s39_40_s59s66.meas[i].meas_label}:")
         Meas_s39_40_s59s66.meas[i].plotTLCC(
         [
             "tbss",
             "segmentation_delays_wo_scheduling_delay",
         ],  # attrbutes name of Meas.delays
         ["TBS", "SegmentDelay(noSched)"],  # labels to display (optional)
     )
```

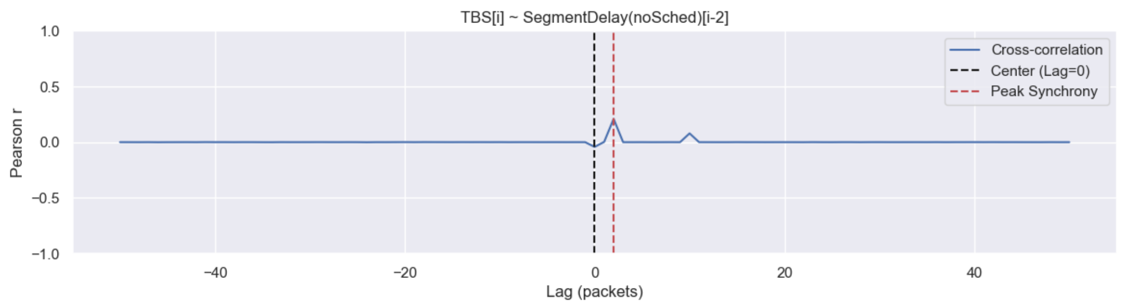
```
s39:
```



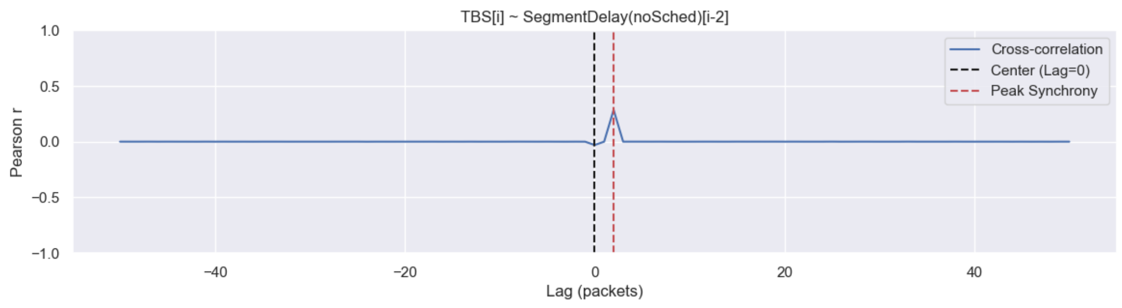
```
s40:
```


TBS[i] ~ SegmentDelay(noSched)[i]
Pearson r
Lag (packets)
Cross-correlation
Center (Lag=0)
Peak Synchrony

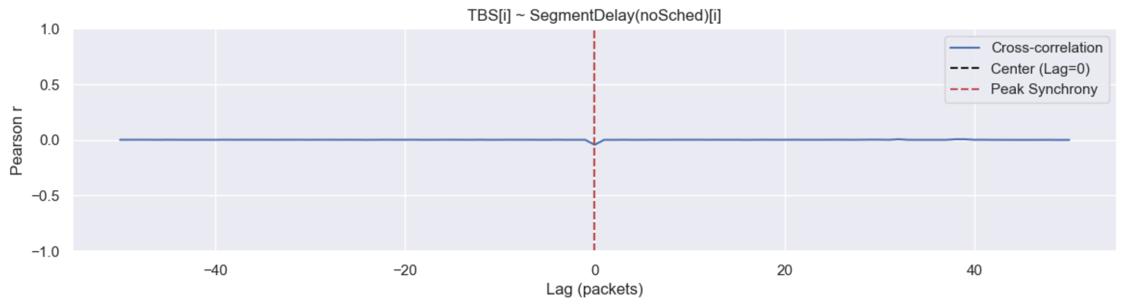
s59:


TBS[i] ~ SegmentDelay(noSched)[i-2]
Pearson r
Lag (packets)
Cross-correlation
Center (Lag=0)
Peak Synchrony

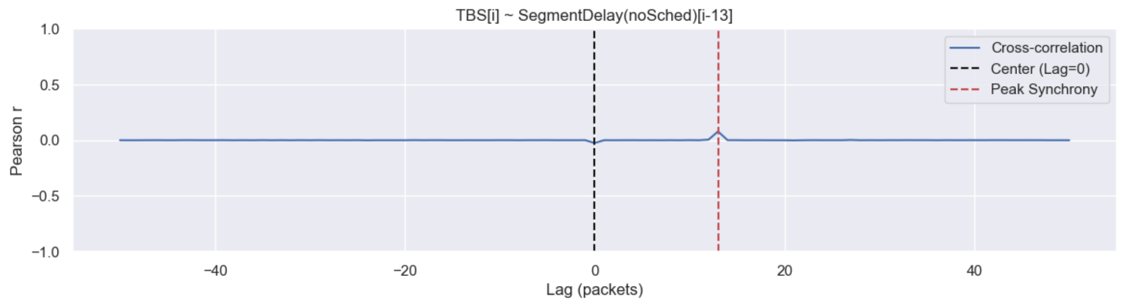
s61:


TBS[i] ~ SegmentDelay(noSched)[i-2]
Pearson r
Lag (packets)
Cross-correlation
Center (Lag=0)
Peak Synchrony

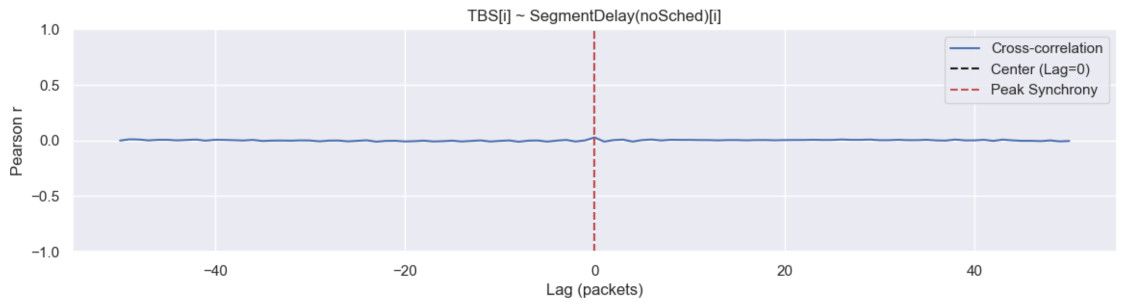
s62:


TBS[i] ~ SegmentDelay(noSched)[i]
Cross-correlation
Center (Lag=0)
Peak Synchrony
Pearson r
Lag (packets)

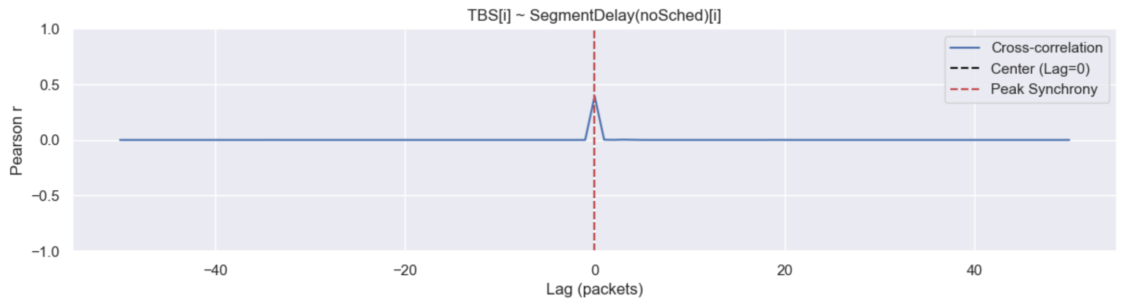
s63:


TBS[i] ~ SegmentDelay(noSched)[i-13]
Cross-correlation
Center (Lag=0)
Peak Synchrony
Pearson r
Lag (packets)

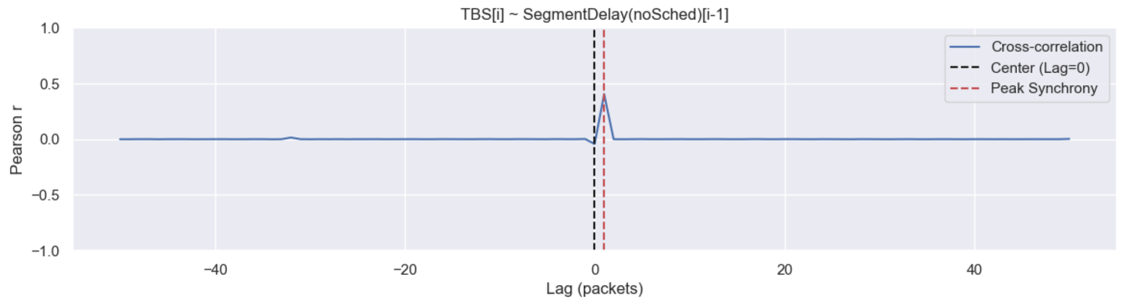
s64:


TBS[i] ~ SegmentDelay(noSched)[i]
Cross-correlation
Center (Lag=0)
Peak Synchrony
Pearson r
Lag (packets)

s65:



```
s66:
```

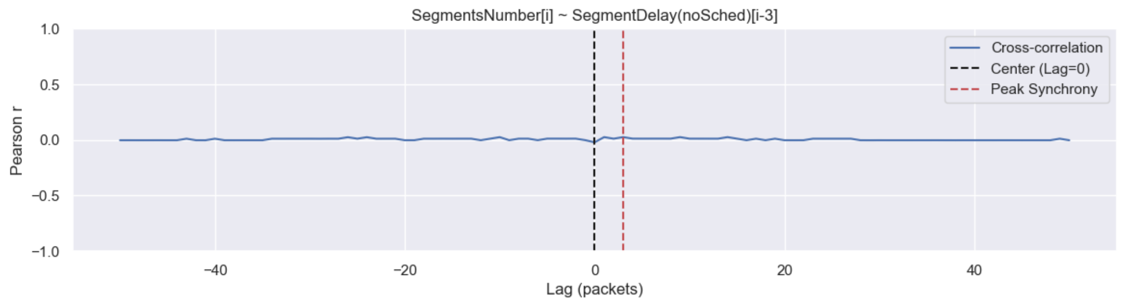


```
s39 s40 s61
segmentation_delays_wo_scheduling_framealignment_delay not found
segmentation_delays_wo_scheduling_framealignment_delay not found
segmentation_delays_wo_scheduling_framealignment_delay not found
```

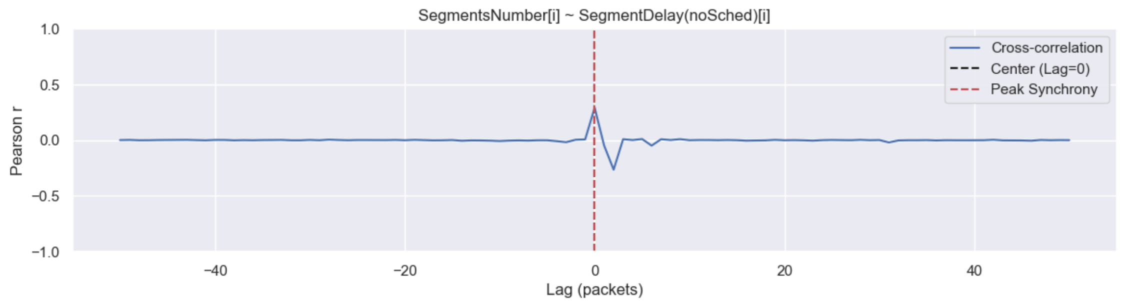
SegmentNum vs SegmentDelay: Should exclude s39, s64 and s65!

[ ]:
```
for i in range(0, len(Meas_s39_40_s59s66.meas_labels)):
    print(f"{Meas_s39_40_s59s66.meas[i].meas_label}:")
    Meas_s39_40_s59s66.meas[i].plotTLCC(
        ["segments", "segmentation_delays_wo_scheduling_delay"],  # attrbutes␣
 ↪name of Meas.delays
        ["SegmentsNumber", "SegmentDelay(noSched)"],  # labels to display␣
 ↪(optional)
    )
```

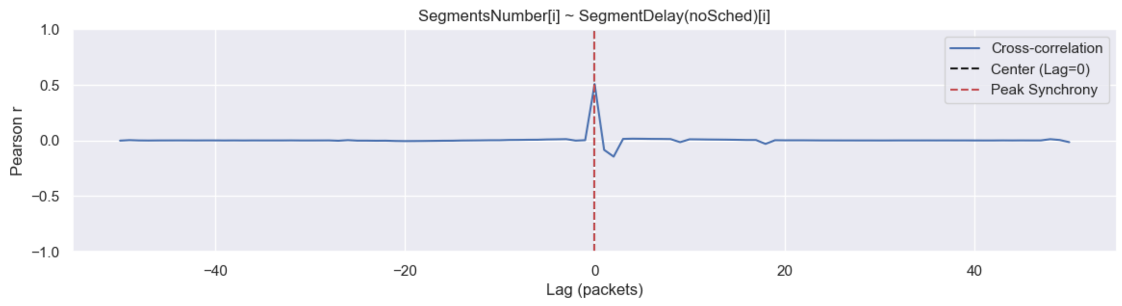
```
s39:
```


SegmentsNumber[i] ~ SegmentDelay(noSched)[i-3]
Cross-correlation
Center (Lag=0)
Peak Synchrony
Pearson r
Lag (packets)

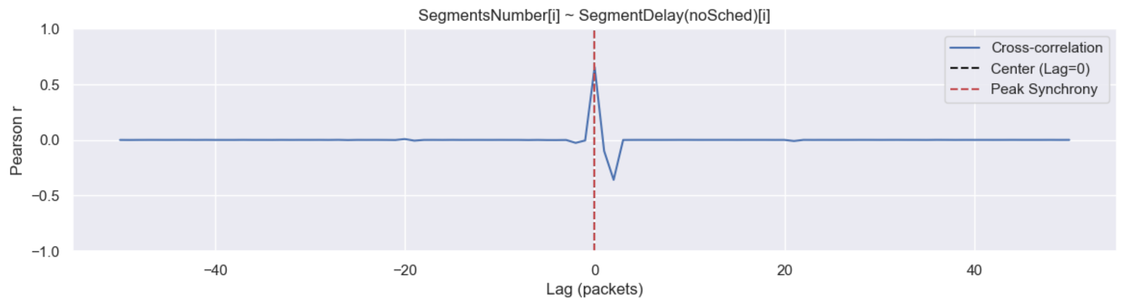
s40:


SegmentsNumber[i] ~ SegmentDelay(noSched)[i]
Cross-correlation
Center (Lag=0)
Peak Synchrony
Pearson r
Lag (packets)

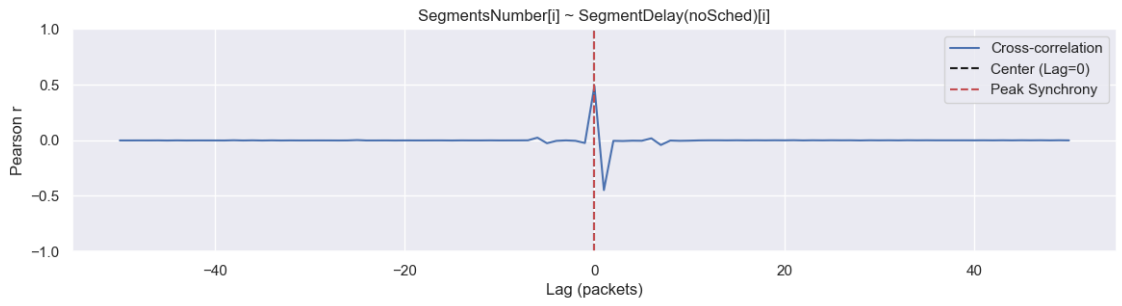
s59:


SegmentsNumber[i] ~ SegmentDelay(noSched)[i]
Cross-correlation
Center (Lag=0)
Peak Synchrony
Pearson r
Lag (packets)

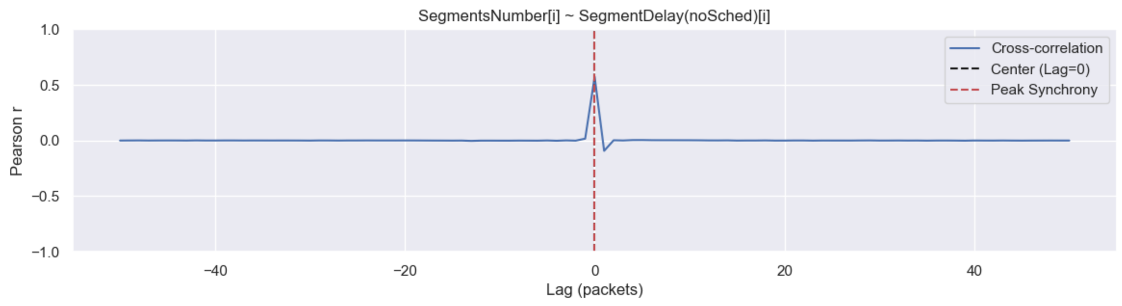
s61:


SegmentsNumber[i] ~ SegmentDelay(noSched)[i]
Pearson r
Lag (packets)
Cross-correlation
Center (Lag=0)
Peak Synchrony

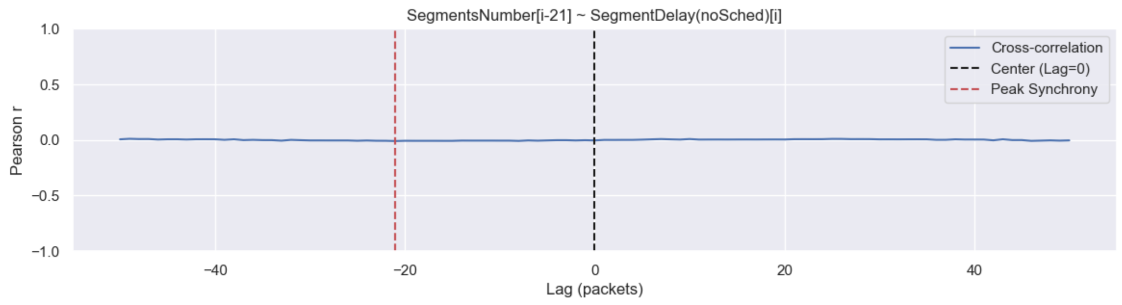
s62:


SegmentsNumber[i] ~ SegmentDelay(noSched)[i]
Pearson r
Lag (packets)
Cross-correlation
Center (Lag=0)
Peak Synchrony

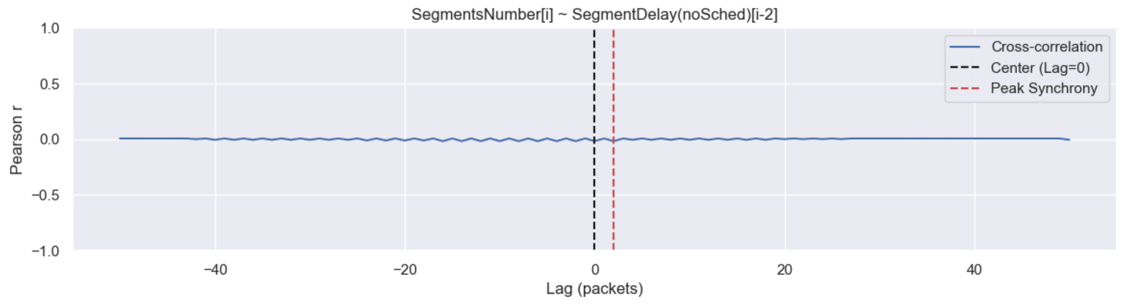
s63:


SegmentsNumber[i] ~ SegmentDelay(noSched)[i]
Pearson r
Lag (packets)
Cross-correlation
Center (Lag=0)
Peak Synchrony

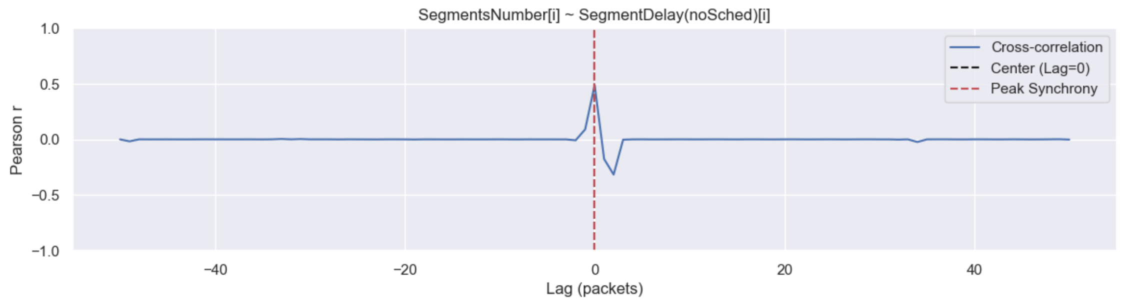
s64:



```
s65:
```



```
s66:
```



```
s65 s64
segmentation_delays_wo_scheduling_framealignment_delay not found
segmentation_delays_wo_scheduling_framealignment_delay not found
```

3 8 “s40”, “s61”,“s62”,“s63”,“s66”

```
[56]: Meas_s40_s616263_s66 = MultiMeas(meas_labels=["s40", "s61", "s62", "s63",␣
        ↪"s66"])
```

```
RNTIs in packets of s40: ['9afe']
RNTIs in packets of s61: ['a431']
2024-12-19 16:01:54.355 | ERROR    |
decomp:get_tx_delay:246 - Packet 29139

phy.in_t or phy.in_t not present
2024-12-19 16:01:54.606 | ERROR    |
decomp:get_tx_delay:246 - Packet 20788

phy.in_t or phy.in_t not present
2024-12-19 16:01:54.841 | ERROR    |
decomp:get_tx_delay:246 - Packet 11082

phy.in_t or phy.in_t not present
2024-12-19 16:01:54.843 | ERROR    |
decomp:get_tx_delay:246 - Packet 11062

phy.in_t or phy.in_t not present
2024-12-19 16:01:55.328 | ERROR    |
decomp:get_tx_delay:246 - Packet 29139

phy.in_t or phy.in_t not present
2024-12-19 16:01:55.482 | ERROR    |
decomp:get_tx_delay:246 - Packet 20788

phy.in_t or phy.in_t not present
2024-12-19 16:01:55.689 | ERROR    |
decomp:get_tx_delay:246 - Packet 11082

phy.in_t or phy.in_t not present
2024-12-19 16:01:55.690 | ERROR    |
decomp:get_tx_delay:246 - Packet 11062

phy.in_t or phy.in_t not present
2024-12-19 16:01:56.284 | ERROR    |
decomp:get_tx_delay:246 - Packet 29139

phy.in_t or phy.in_t not present
2024-12-19 16:01:56.526 | ERROR    |
decomp:get_tx_delay:246 - Packet 20788

phy.in_t or phy.in_t not present
2024-12-19 16:01:56.821 | ERROR    |
decomp:get_tx_delay:246 - Packet 11082

phy.in_t or phy.in_t not present
2024-12-19 16:01:56.822 | ERROR    |
decomp:get_tx_delay:246 - Packet 11062

phy.in_t or phy.in_t not present
2024-12-19 16:01:59.640 | ERROR    |
decomp:get_tx_delay:246 - Packet 29139

phy.in_t or phy.in_t not present
2024-12-19 16:01:59.755 | ERROR    |
decomp:get_tx_delay:246 - Packet 20788

phy.in_t or phy.in_t not present
2024-12-19 16:01:59.904 | ERROR    |
decomp:get_tx_delay:246 - Packet 11082

phy.in_t or phy.in_t not present
2024-12-19 16:01:59.905 | ERROR    |
decomp:get_tx_delay:246 - Packet 11062

phy.in_t or phy.in_t not present
2024-12-19 16:02:00.399 | ERROR    |
decomp:get_tx_delay:246 - Packet 29139

phy.in_t or phy.in_t not present
2024-12-19 16:02:00.721 | ERROR    |
decomp:get_tx_delay:246 - Packet 20788

phy.in_t or phy.in_t not present
2024-12-19 16:02:01.053 | ERROR    |
decomp:get_tx_delay:246 - Packet 11082

phy.in_t or phy.in_t not present
2024-12-19 16:02:01.055 | ERROR    |
decomp:get_tx_delay:246 - Packet 11062

phy.in_t or phy.in_t not present
2024-12-19 16:02:02.006 | ERROR    |
decomp:get_tx_delay:246 - Packet 29139

phy.in_t or phy.in_t not present
2024-12-19 16:02:02.007 | ERROR    |
decomp:get_tx_delay:246 - Packet 29139

phy.in_t or phy.in_t not present
2024-12-19 16:02:02.335 | ERROR    |
decomp:get_tx_delay:246 - Packet 20788

phy.in_t or phy.in_t not present
2024-12-19 16:02:02.337 | ERROR    |
decomp:get_tx_delay:246 - Packet 20788

phy.in_t or phy.in_t not present
2024-12-19 16:02:02.720 | ERROR    |
decomp:get_tx_delay:246 - Packet 24537

phy.in_t or phy.in_t not present
2024-12-19 16:03:24.528 | ERROR    |
decomp:get_tx_delay:246 - Packet 24537

phy.in_t or phy.in_t not present
2024-12-19 16:03:24.606 | ERROR    |
decomp:get_tx_delay:246 - Packet 15841

phy.in_t or phy.in_t not present
2024-12-19 16:03:24.607 | ERROR    |
decomp:get_tx_delay:246 - Packet 15841

phy.in_t or phy.in_t not present
2024-12-19 16:03:24.686 | ERROR    |
decomp:get_tx_delay:246 - Packet 7385

phy.in_t or phy.in_t not present
2024-12-19 16:03:24.687 | ERROR    |
decomp:get_tx_delay:246 - Packet 7385

phy.in_t or phy.in_t not present
2024-12-19 16:03:24.746 | ERROR    |
decomp:get_tx_delay:246 - Packet 3730

phy.in_t or phy.in_t not present
2024-12-19 16:03:24.750 | ERROR    |
decomp:get_tx_delay:246 - Packet 3730

phy.in_t or phy.in_t not present
2024-12-19 16:03:24.752 | ERROR    |
decomp:get_tx_delay:246 - Packet 3729

phy.in_t or phy.in_t not present
2024-12-19 16:03:24.754 | ERROR    |
decomp:get_tx_delay:246 - Packet 3729

phy.in_t or phy.in_t not present
2024-12-19 16:03:24.755 | ERROR    |
decomp:get_tx_delay:246 - Packet 3729

phy.in_t or phy.in_t not present
2024-12-19 16:03:24.756 | ERROR    |
decomp:get_tx_delay:246 - Packet 3729

phy.in_t or phy.in_t not present
2024-12-19 16:03:24.794 | ERROR    |
decomp:get_tx_delay:246 - Packet 142

phy.in_t or phy.in_t not present
2024-12-19 16:03:24.796 | ERROR    |
decomp:get_tx_delay:246 - Packet 142

phy.in_t or phy.in_t not present
```

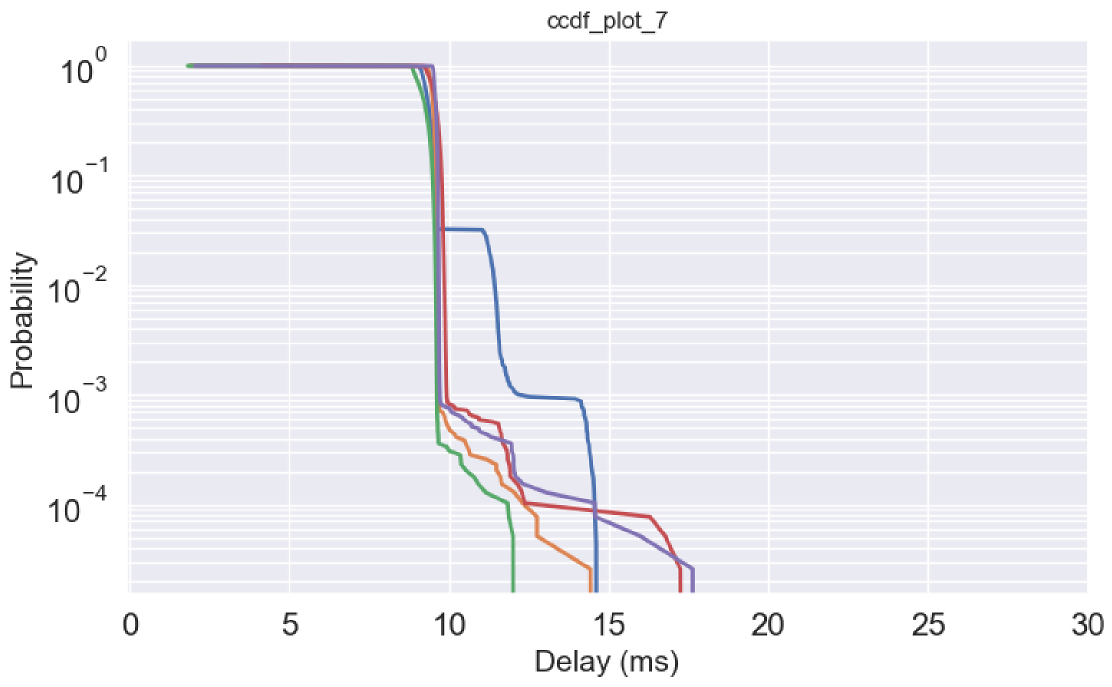
Export csv

```
[57]: Meas_s40_s616263_s66.dataFrame(
          [
              "tbss",
              "segments",
              "segmentation_delays_wo_scheduling_delay",
          ],  # attrbutes name of Meas.delays
          [
              "TBS",
              "SegmentsNum",
              "SegmentDelay(noSched)",
          ],  # labels to display (optional)
      )
```

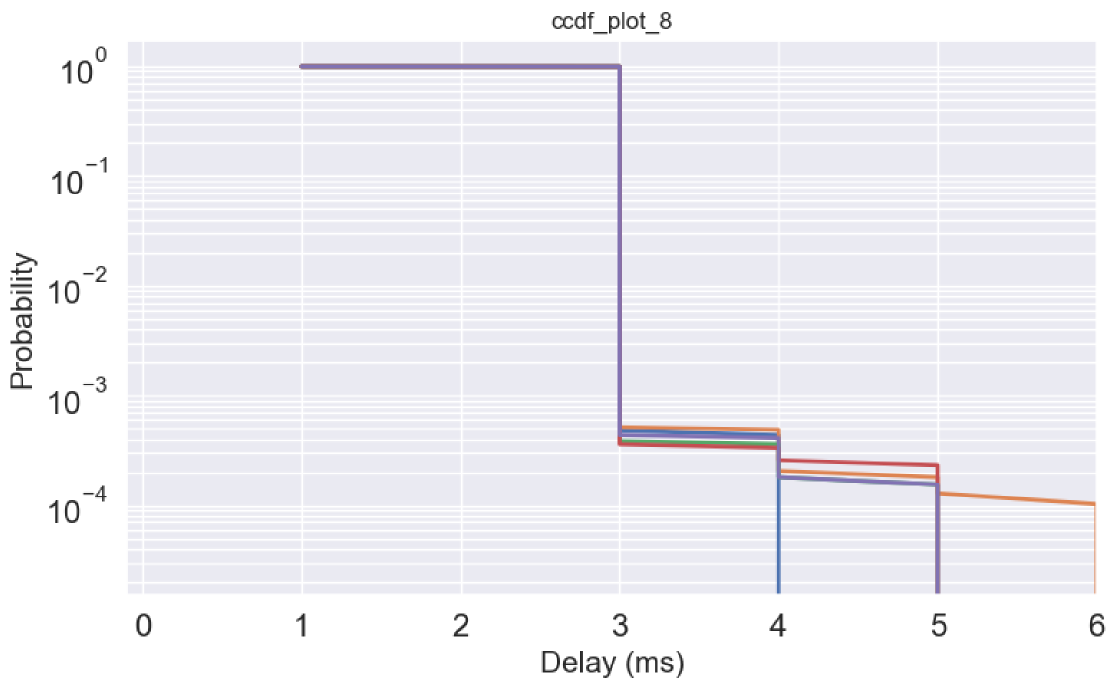
```
   TBS  SegmentsNum  SegmentDelay(noSched)
0   24            4              11.732101
1  116            1               2.104998
2   24            4              12.003183
3   24            3               9.509802
4   24            1               9.574890
./data/csv/
Dataframe saved to ./data/csv/s40_TBS_SegmentsNum_SegmentDelay(noSched).csv
   TBS  SegmentsNum  SegmentDelay(noSched)
0   24            1               9.579897
1   24            3               9.560108
2   24            1               9.532213
3   24            3               9.479761
4   24            2              12.001991
Dataframe saved to ./data/csv/s61_TBS_SegmentsNum_SegmentDelay(noSched).csv
   TBS  SegmentsNum  SegmentDelay(noSched)
0   24            3               9.410858
1   24            3               9.418011
2   24            4              11.901855
3   24            4              11.814833
4   24            3               9.358883
Dataframe saved to ./data/csv/s62_TBS_SegmentsNum_SegmentDelay(noSched).csv
   TBS  SegmentsNum  SegmentDelay(noSched)
0   24            1               9.531975
1   24            3               9.627819
2   24            1               9.535789
3   24            3               9.494066
4   24            2              11.957884
Dataframe saved to ./data/csv/s63_TBS_SegmentsNum_SegmentDelay(noSched).csv
   TBS  SegmentsNum  SegmentDelay(noSched)
```

```
0   24            1                9.522915
1   24            3                9.600878
2   24            2               12.004852
3   24            3                9.702682
4   24            1                9.557724
Dataframe saved to ./data/csv/s66_TBS_SegmentsNum_SegmentDelay(noSched).csv
```

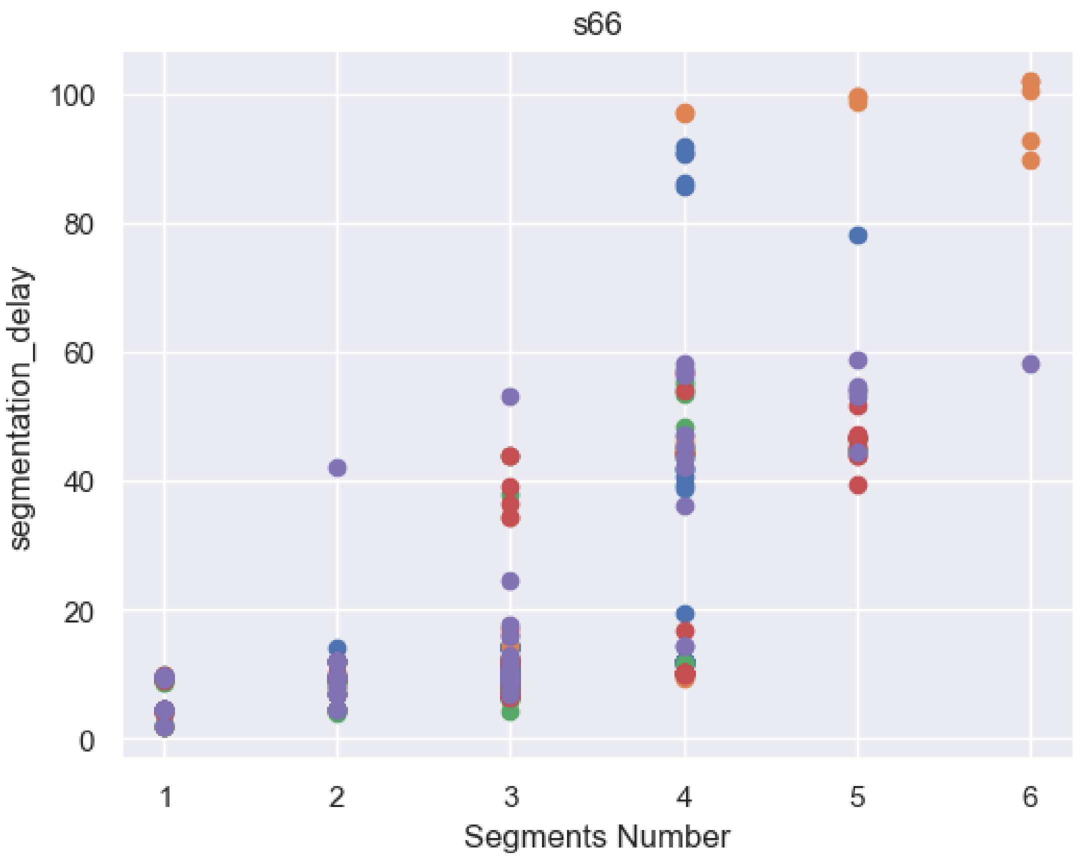
[58]:
```
Meas_s40_s616263_s66.plotCCDF(
    ["segmentation_delays_wo_scheduling_delay"],
    ["s40", "s61", "s62", "s63", "s66"],
)
```



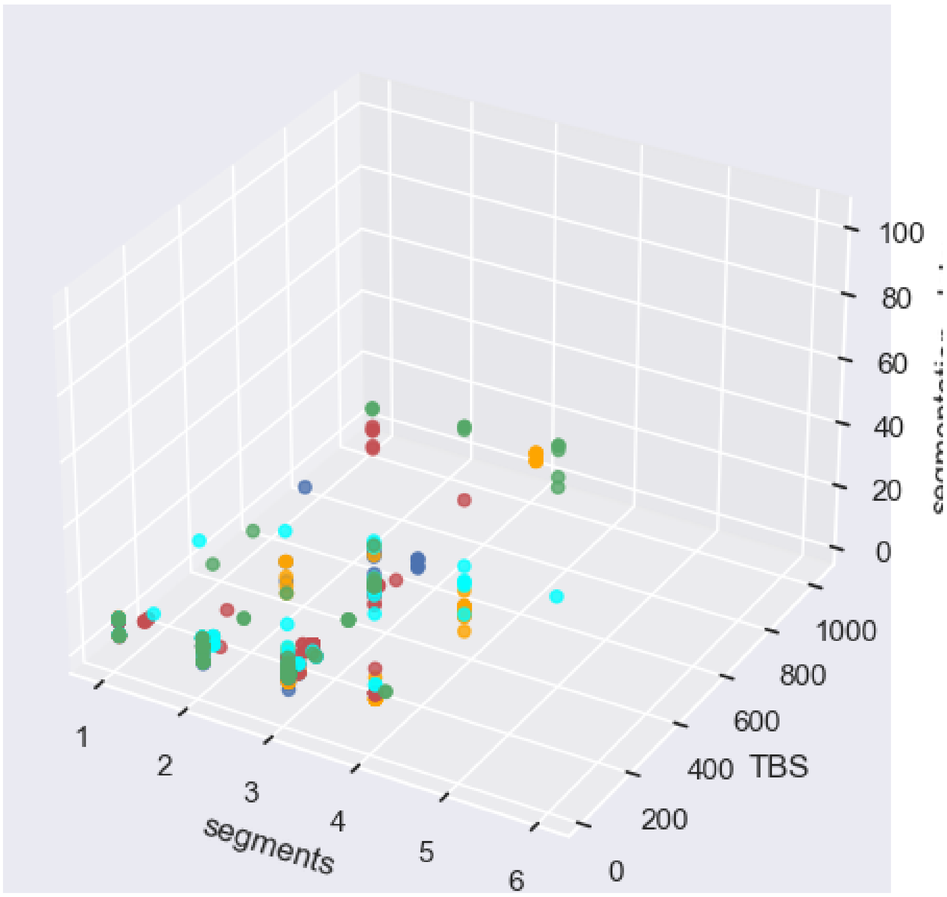
[59]:
```
Meas_s40_s616263_s66.plotCCDF(
    ["segments"],
    ["s40", "s61", "s62", "s63", "s66"],
)
```



```
[60]: for i in range(0,5):
          x = Meas_s40_s616263_s66.meas[i].delays.segments
          y = Meas_s40_s616263_s66.meas[i].delays.
       ↪segmentation_delays_wo_scheduling_delay
          plt.scatter(x,y)
          plt.xlabel("Segments Number")
          plt.ylabel("segmentation_delay")
          plt.title(f"{Meas_s40_s616263_s66.meas[i].meas_label}")
```



[61]:
```
fig = plt.figure(figsize=(8, 6))
#     3D
ax = fig.add_subplot(111, projection="3d")
c=["r","g","b","orange","cyan","m"]
for i in range(0, 5):
    x = Meas_s40_s616263_s66.meas[i].delays.segments
    y = Meas_s40_s616263_s66.meas[i].delays.tbss
    z = Meas_s40_s616263_s66.meas[i].delays.
 ↪segmentation_delays_wo_scheduling_delay
    #   3D
    ax.scatter(x, y, z, c=c[i], marker="o", alpha=0.8)
    #

#
ax.set_xlabel("segments")
ax.set_ylabel("TBS")
ax.set_zlabel("segmentation_delay")

plt.show()
```



4 9 “s40”, “s59”, “s61”,“s62”,“s63”,“s66”

```
[62]: Meas_s40_s59_s616263_s66 = MultiMeas(meas_labels=["s40", "s59", "s61", "s62",␣
        ↪"s63", "s66"])
```

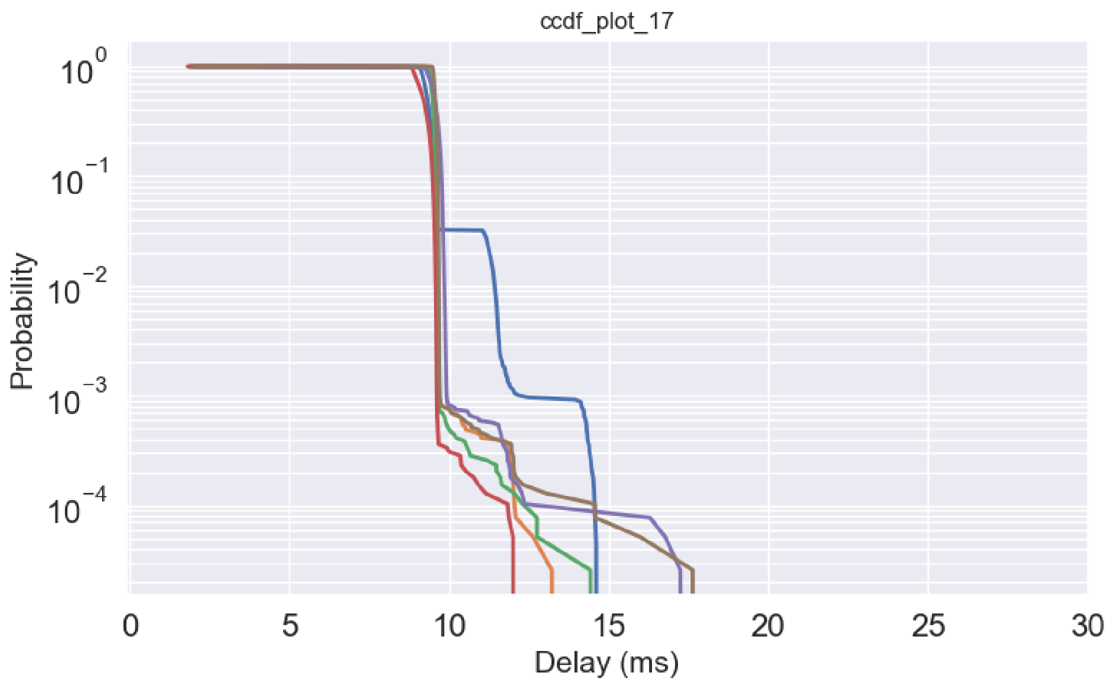
```
RNTIs in packets of s40: ['9afe']
RNTIs in packets of s59: ['7b9c']
2024-12-19 16:05:40.136 | ERROR    |
decomp:get_tx_delay:246 - Packet 37991

phy.in_t or phy.in_t not present
2024-12-19 16:05:40.189 | ERROR    |
decomp:get_tx_delay:246 - Packet 33786

phy.in_t or phy.in_t not present
2024-12-19 16:05:40.190 | ERROR    |
decomp:get_tx_delay:246 - Packet 33786

phy.in_t or phy.in_t not present
```

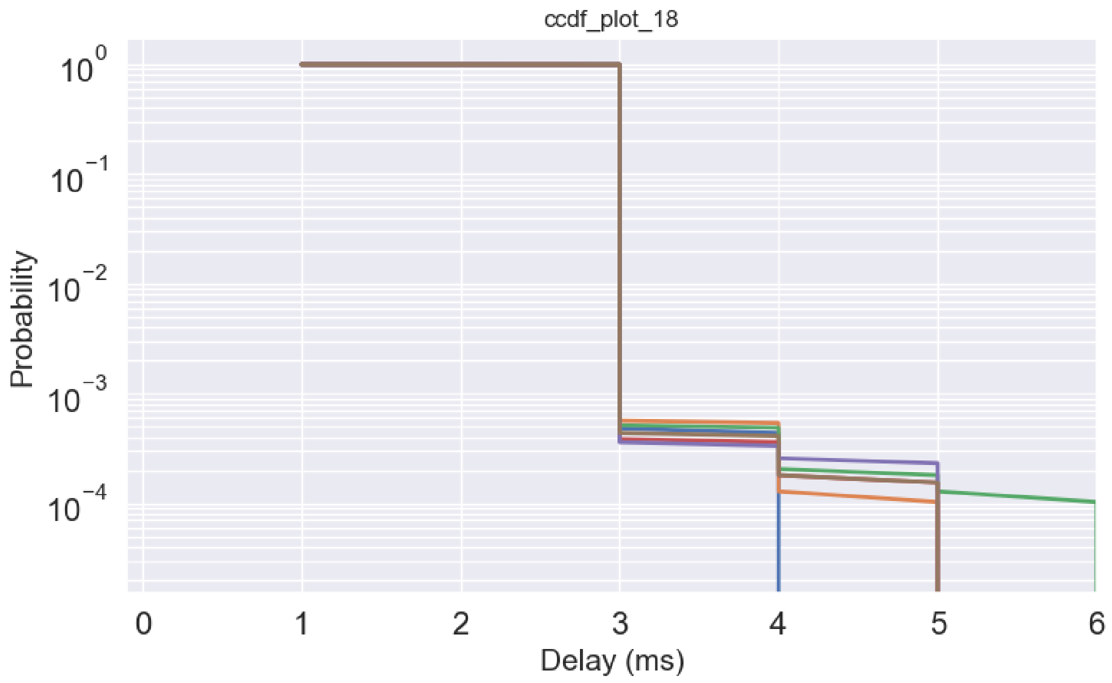
```
2024-12-19 16:05:40.191 | ERROR    |
decomp:get_tx_delay:246 - Packet 33786
phy.in_t or phy.in_t not present
2024-12-19 16:05:40.191 | ERROR    |
decomp:get_tx_delay:246 - Packet 33786
phy.in_t or phy.in_t not present
2024-12-19 16:05:40.219 | ERROR    |
decomp:get_tx_delay:246 - Packet 31778
phy.in_t or phy.in_t not present
2024-12-19 16:05:40.220 | ERROR    |
decomp:get_tx_delay:246 - Packet 31778
phy.in_t or phy.in_t not present
2024-12-19 16:05:40.221 | ERROR    |
decomp:get_tx_delay:246 - Packet 31778
phy.in_t or phy.in_t not present
2024-12-19 16:05:40.222 | ERROR    |
decomp:get_tx_delay:246 - Packet 31778
phy.in_t or phy.in_t not present
2024-12-19 16:05:40.223 | ERROR    |
decomp:get_tx_delay:246 - Packet 31778
phy.in_t or phy.in_t not present
2024-12-19 16:05:40.224 | ERROR    |
decomp:get_tx_delay:246 - Packet 31778
phy.in_t or phy.in_t not present
2024-12-19 16:05:40.227 | ERROR    |
decomp:get_tx_delay:246 - Packet 31778
phy.in_t or phy.in_t not present
2024-12-19 16:05:40.227 | ERROR    |
decomp:get_tx_delay:246 - Packet 31778
phy.in_t or phy.in_t not present
2024-12-19 16:05:40.243 | ERROR    |
decomp:get_tx_delay:246 - Packet 30777
phy.in_t or phy.in_t not present
2024-12-19 16:05:40.244 | ERROR    |
decomp:get_tx_delay:246 - Packet 30777
phy.in_t or phy.in_t not present
2024-12-19 16:05:40.244 | ERROR    |
decomp:get_tx_delay:246 - Packet 30777
phy.in_t or phy.in_t not present
2024-12-19 16:05:40.274 | ERROR    |
decomp:get_tx_delay:246 - Packet 15841

phy.in_t or phy.in_t not present
2024-12-19 16:07:23.605 | ERROR    |
decomp:get_tx_delay:246 - Packet 15841

phy.in_t or phy.in_t not present
2024-12-19 16:07:23.686 | ERROR    |
decomp:get_tx_delay:246 - Packet 7385

phy.in_t or phy.in_t not present
2024-12-19 16:07:23.687 | ERROR    |
decomp:get_tx_delay:246 - Packet 7385

phy.in_t or phy.in_t not present
2024-12-19 16:07:23.721 | ERROR    |
decomp:get_tx_delay:246 - Packet 3730

phy.in_t or phy.in_t not present
2024-12-19 16:07:23.722 | ERROR    |
decomp:get_tx_delay:246 - Packet 3730

phy.in_t or phy.in_t not present
2024-12-19 16:07:23.723 | ERROR    |
decomp:get_tx_delay:246 - Packet 3729

phy.in_t or phy.in_t not present
2024-12-19 16:07:23.724 | ERROR    |
decomp:get_tx_delay:246 - Packet 3729

phy.in_t or phy.in_t not present
2024-12-19 16:07:23.725 | ERROR    |
decomp:get_tx_delay:246 - Packet 3729

phy.in_t or phy.in_t not present
2024-12-19 16:07:23.727 | ERROR    |
decomp:get_tx_delay:246 - Packet 3729

phy.in_t or phy.in_t not present
2024-12-19 16:07:23.761 | ERROR    |
decomp:get_tx_delay:246 - Packet 142

phy.in_t or phy.in_t not present
2024-12-19 16:07:23.762 | ERROR    |
decomp:get_tx_delay:246 - Packet 142

phy.in_t or phy.in_t not present
```

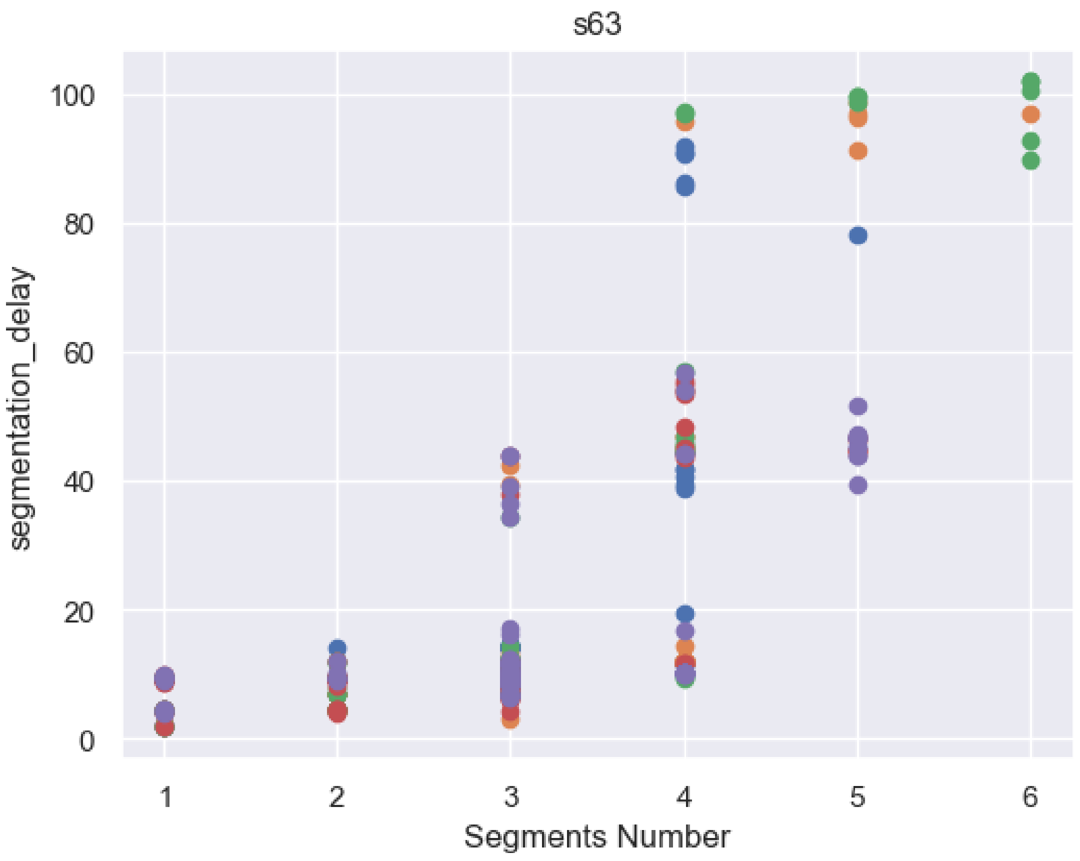
[63]:
```
Meas_s40_s59_s616263_s66.dataFrame(
    [
        "tbss",
        "segments",
        "segmentation_delays_wo_scheduling_delay",
    ],  # attrbutes name of Meas.delays
    [
        "TBS",
        "SegmentsNum",
        "SegmentDelay(noSched)",
    ],  # labels to display (optional)
)
Meas_s40_s59_s616263_s66.plotCCDF(
    ["segmentation_delays_wo_scheduling_delay"],
    ["s40", "s59", "s61", "s62", "s63", "s66"],
)
Meas_s40_s59_s616263_s66.plotCCDF(
    ["segments"],
    ["s40", "s59", "s61", "s62", "s63", "s66"],
)
for i in range(0, 5):
    x = Meas_s40_s59_s616263_s66.meas[i].delays.segments
    y = Meas_s40_s59_s616263_s66.meas[i].delays.
 ↪segmentation_delays_wo_scheduling_delay
    plt.scatter(x, y)
    plt.xlabel("Segments Number")
    plt.ylabel("segmentation_delay")
    plt.title(f"{Meas_s40_s59_s616263_s66.meas[i].meas_label}")
plt.show()

fig = plt.figure(figsize=(8, 6))
#     3D
ax = fig.add_subplot(111, projection="3d")
#c = ["r", "g", "b", "orange", "cyan", "m"]
for i in range(0, 6):
    x = Meas_s40_s59_s616263_s66.meas[i].delays.segments
    y = Meas_s40_s59_s616263_s66.meas[i].delays.tbss
    z = Meas_s40_s59_s616263_s66.meas[i].delays.
 ↪segmentation_delays_wo_scheduling_delay
    #   3D
    ax.scatter(x, y, z, marker="o", alpha=0.8)
    #

#
ax.set_xlabel("segments")
ax.set_ylabel("TBS")
ax.set_zlabel("segmentation_delay")
plt.show()
```

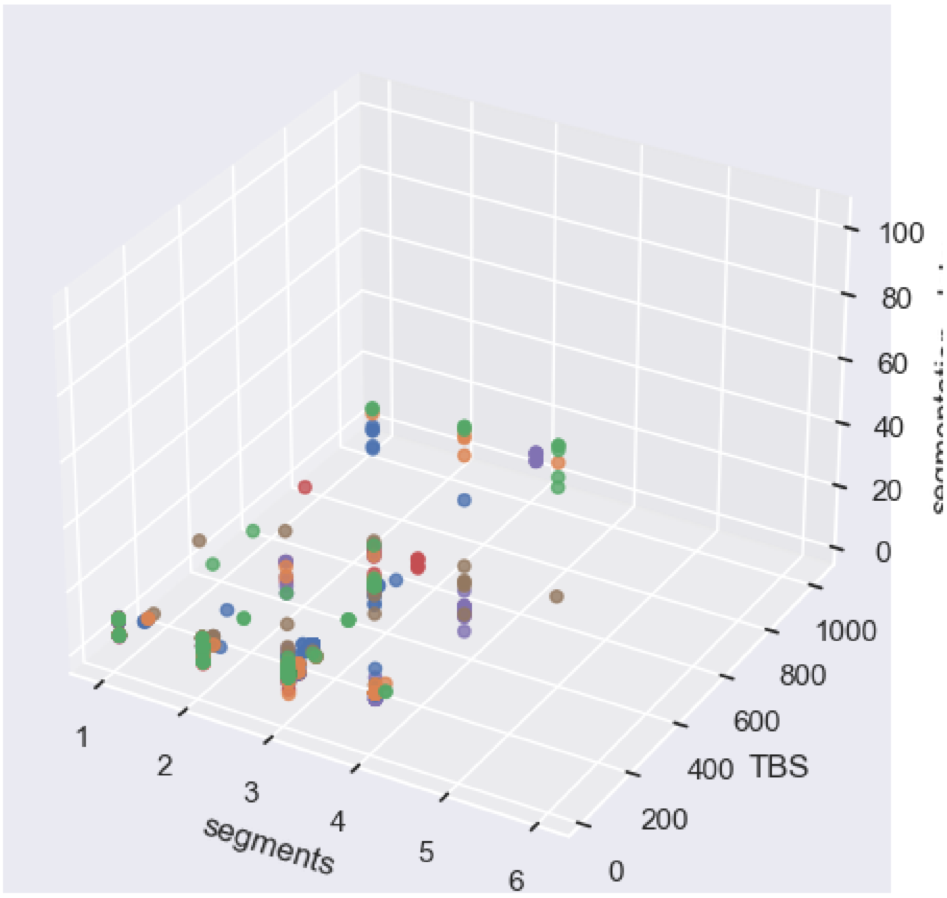
```
   TBS  SegmentsNum  SegmentDelay(noSched)
0   24            4              11.732101
1  116            1               2.104998
```

```
2   24           4             12.003183
3   24           3              9.509802
4   24           1              9.574890
./data/csv/
Dataframe saved to ./data/csv/s40_TBS_SegmentsNum_SegmentDelay(noSched).csv
   TBS  SegmentsNum  SegmentDelay(noSched)
0   24           1              9.505987
1   24           3              9.545088
2   24           2             11.962891
3   24           3              9.494781
4   24           1              9.520769
Dataframe saved to ./data/csv/s59_TBS_SegmentsNum_SegmentDelay(noSched).csv
   TBS  SegmentsNum  SegmentDelay(noSched)
0   24           1              9.579897
1   24           3              9.560108
2   24           1              9.532213
3   24           3              9.479761
4   24           2             12.001991
Dataframe saved to ./data/csv/s61_TBS_SegmentsNum_SegmentDelay(noSched).csv
   TBS  SegmentsNum  SegmentDelay(noSched)
0   24           3              9.410858
1   24           3              9.418011
2   24           4             11.901855
3   24           4             11.814833
4   24           3              9.358883
Dataframe saved to ./data/csv/s62_TBS_SegmentsNum_SegmentDelay(noSched).csv
   TBS  SegmentsNum  SegmentDelay(noSched)
0   24           1              9.531975
1   24           3              9.627819
2   24           1              9.535789
3   24           3              9.494066
4   24           2             11.957884
Dataframe saved to ./data/csv/s63_TBS_SegmentsNum_SegmentDelay(noSched).csv
   TBS  SegmentsNum  SegmentDelay(noSched)
0   24           1              9.522915
1   24           3              9.600878
2   24           2             12.004852
3   24           3              9.702682
4   24           1              9.557724
Dataframe saved to ./data/csv/s66_TBS_SegmentsNum_SegmentDelay(noSched).csv
```


ccdf_plot_17
Probability
Delay (ms)
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
5
10
15
20
25
30


ccdf_plot_18
Probability
Delay (ms)
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
1
2
3
4
5
6


s63
segmentation_delay
0
20
40
60
80
100
1
2
3
4
5
6
Segments Number



[64]:
```
import glob
print("Reading the CSV files")
csv_files = glob.glob(f"data/csv/*.csv")
print("\nCombining all uploaded files into a single DataFrame...")
combined_df = pd.concat(
    [pd.read_csv(file, index_col=0) for file in csv_files], ignore_index=True
)
df = combined_df
```

```
Reading the CSV files

Combining all uploaded files into a single DataFrame…
```

[65]:
```
len(df)
```

[65]:
```
223854
```

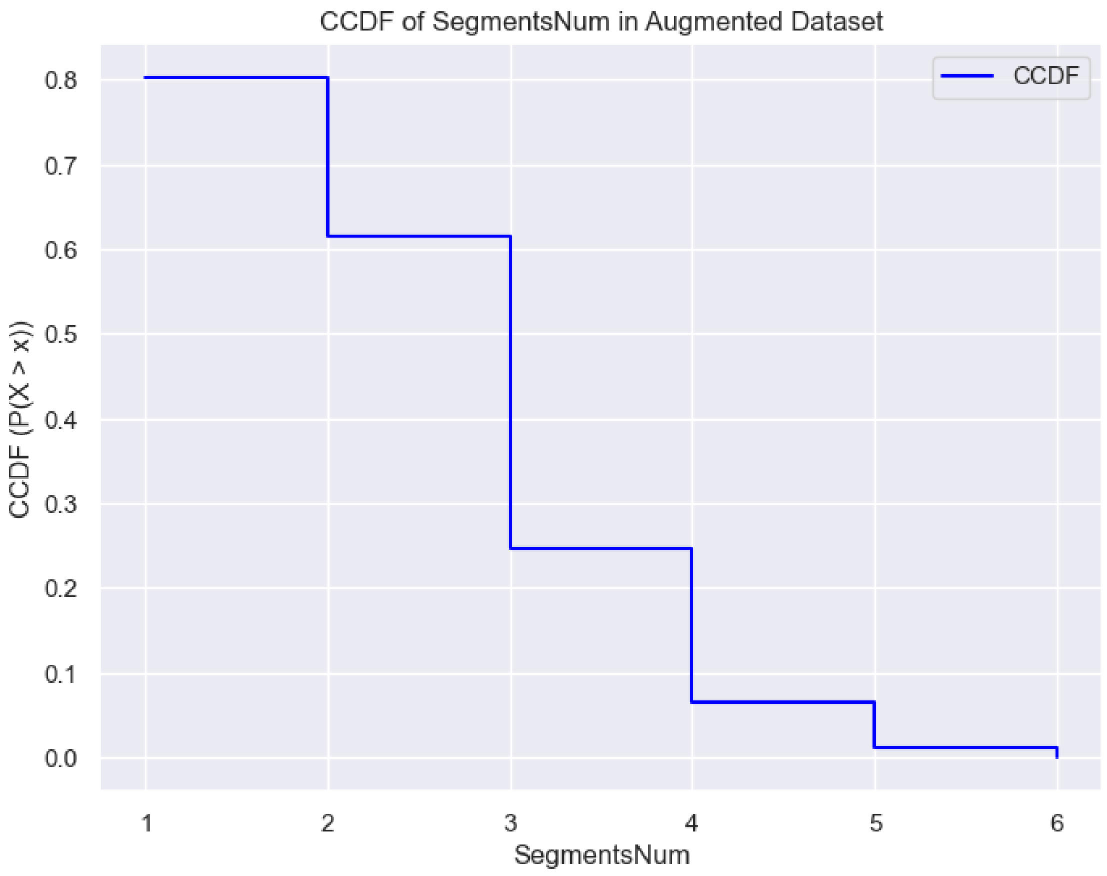
[66]:
```python
import pandas as pd
import numpy as np
from tqdm import tqdm

# Example: df is your original dataframe
# df must contain columns: "SegmentsNum", "TBS", "SegmentDelay(noSched)"
# Ensure df has no missing values or handle them as needed

# Filter rows where SegmentsNum >= 3
high_delay_rows = df[df["SegmentsNum"] != 3]

# List to hold augmented data
augmented_data = []

# Number of duplicates per row
N_DUPLICATES = 1000

for idx, row in tqdm(
    high_delay_rows.iterrows(),
    total=high_delay_rows.shape[0],
    desc="Augmenting Data",
    unit="row",
):
    original_delay = row["SegmentDelay(noSched)"]

    # Calculate standard deviation for the Gaussian noise
    # If original_delay can be zero or negative, handle that case appropriately
    # For now, we assume it's positive
    sigma = 0.0001 * original_delay

    # Generate noise and add to the original SegmentDelay(noSched)
    noise = np.random.randn(N_DUPLICATES) * sigma
    new_delays = original_delay + noise

    # Create a DataFrame of duplicated rows
    # Keep all columns the same except SegmentDelay(noSched), which will be␣
 ↪varied
    replicated_rows = pd.DataFrame(
        {
            "SegmentsNum": np.repeat(row["SegmentsNum"], N_DUPLICATES),
            "TBS": np.repeat(row["TBS"], N_DUPLICATES),
            "SegmentDelay(noSched)": new_delays,
        }
    )

    # If there are other columns in df that you want to keep as is:
```

```
    # For example, if df has columns ["SegmentsNum", "TBS",␣
↪"SegmentDelay(noSched)", "SomeOtherFeat"]
    # replicated_rows = pd.concat([
    #     pd.DataFrame(np.repeat([row.drop("SegmentDelay(noSched)").values],␣
↪N_DUPLICATES, axis=0),
    #                  columns=row.drop("SegmentDelay(noSched)").index),
    #     pd.DataFrame({"SegmentDelay(noSched)": new_delays})
    # ], axis=1)

    augmented_data.append(replicated_rows)

# Combine all augmented rows
augmented_df = pd.concat(augmented_data, ignore_index=True)

# Append to original dataframe (optional)
df_augmented = pd.concat([df, augmented_df], ignore_index=True)

# Now df_augmented contains your original data plus the augmented samples
print(
    "Augmentation complete. Original size:",
    len(df),
    "Augmented size:",
    len(df_augmented),
)
```

```
Augmenting Data: 100%|      | 384/384 [00:00<00:00, 2314.78row/s]

Augmentation complete. Original size: 223854 Augmented size: 607854
```

[67]:
```
# high_delay_rows = df[df["SegmentsNum"] == 4]
# len(high_delay_rows)
```

[68]:
```
import numpy as np
import matplotlib.pyplot as plt

# Extract the segmentNum column
segment_nums = df_augmented["SegmentsNum"].values

# Sort the values in ascending order
sorted_vals = np.sort(segment_nums)
n = len(sorted_vals)

# Get the unique values to plot against
unique_vals = np.unique(sorted_vals)

# Compute the CCDF for each unique value
ccdf = []
for val in unique_vals:
    # CCDF(val) = P(X > val) = number of elements greater than val / total␣
↪elements
    ccdf_val = np.sum(sorted_vals > val) / n
    ccdf.append(ccdf_val)

# Plot the CCDF
plt.figure(figsize=(8, 6))
plt.step(unique_vals, ccdf, where="post", label="CCDF", color="blue")

plt.xlabel("SegmentsNum")
plt.ylabel("CCDF (P(X > x))")
plt.title("CCDF of SegmentsNum in Augmented Dataset")
plt.grid(True)
plt.legend()

# Optional: Use a log scale if desired to highlight tail behavior
# plt.yscale('log')
# plt.xscale('log')

plt.show()
```



5 10 “s40”,“s62”,“s63”,“s66”

[69]:
```
Meas_s40_s6263_s66 = MultiMeas(meas_labels=["s40", "s62", "s63", "s66"])
```

```
RNTIs in packets of s40: ['9afe']
RNTIs in packets of s62: ['a244']
2024-12-19 16:10:20.829 | ERROR    |
decomp:get_tx_delay:246 - Packet 39975

phy.in_t or phy.in_t not present
2024-12-19 16:10:20.839 | ERROR    |
decomp:get_tx_delay:246 - Packet 39851

phy.in_t or phy.in_t not present
2024-12-19 16:10:20.841 | ERROR    |
decomp:get_tx_delay:246 - Packet 39850

phy.in_t or phy.in_t not present
2024-12-19 16:10:20.932 | ERROR    |
decomp:get_tx_delay:246 - Packet 37156

phy.in_t or phy.in_t not present
2024-12-19 16:10:20.933 | ERROR    |
decomp:get_tx_delay:246 - Packet 37149

phy.in_t or phy.in_t not present
2024-12-19 16:10:20.947 | ERROR    |
decomp:get_tx_delay:246 - Packet 36729

phy.in_t or phy.in_t not present
2024-12-19 16:10:21.023 | ERROR    |
decomp:get_tx_delay:246 - Packet 34309

phy.in_t or phy.in_t not present
2024-12-19 16:10:21.047 | ERROR    |
decomp:get_tx_delay:246 - Packet 33978

phy.in_t or phy.in_t not present
2024-12-19 16:10:21.049 | ERROR    |
decomp:get_tx_delay:246 - Packet 33977

phy.in_t or phy.in_t not present
2024-12-19 16:10:21.119 | ERROR    |
decomp:get_tx_delay:246 - Packet 32712

phy.in_t or phy.in_t not present
2024-12-19 16:10:21.250 | ERROR    |
decomp:get_tx_delay:246 - Packet 27083

phy.in_t or phy.in_t not present
```

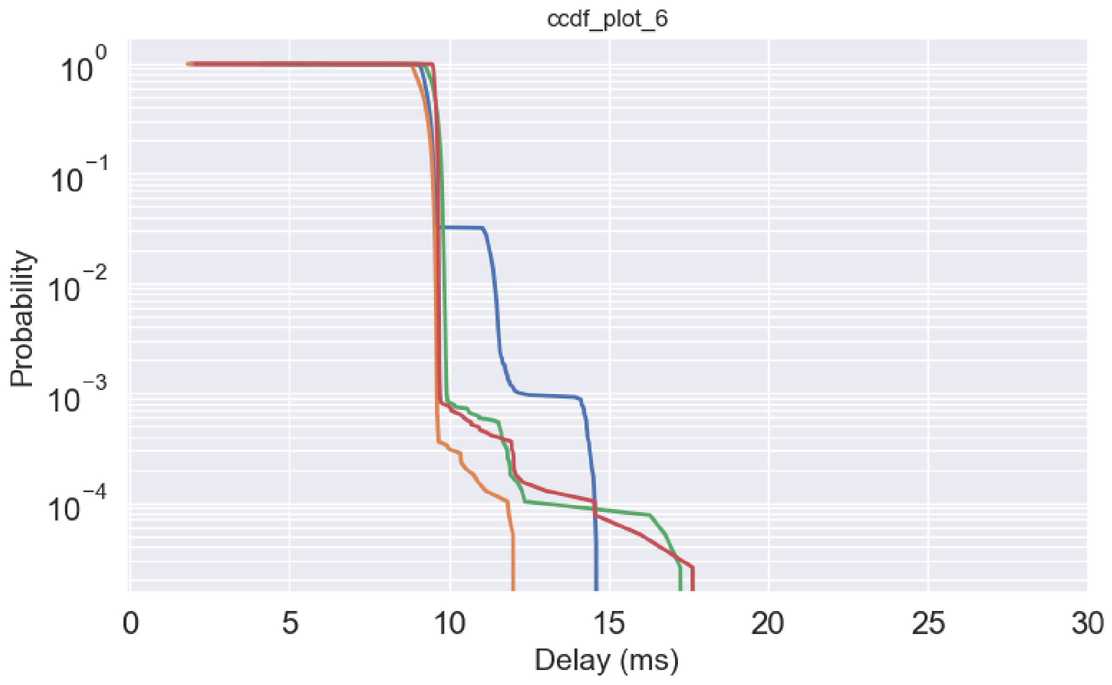
```
decomp:get_tx_delay:246 - Packet 24537

phy.in_t or phy.in_t not present
2024-12-19 16:11:18.363 | ERROR    |
decomp:get_tx_delay:246 - Packet 24537

phy.in_t or phy.in_t not present
2024-12-19 16:11:18.506 | ERROR    |
decomp:get_tx_delay:246 - Packet 15841

phy.in_t or phy.in_t not present
2024-12-19 16:11:18.507 | ERROR    |
decomp:get_tx_delay:246 - Packet 15841

phy.in_t or phy.in_t not present
2024-12-19 16:11:18.615 | ERROR    |
decomp:get_tx_delay:246 - Packet 7385

phy.in_t or phy.in_t not present
2024-12-19 16:11:18.617 | ERROR    |
decomp:get_tx_delay:246 - Packet 7385

phy.in_t or phy.in_t not present
2024-12-19 16:11:18.663 | ERROR    |
decomp:get_tx_delay:246 - Packet 3730

phy.in_t or phy.in_t not present
2024-12-19 16:11:18.664 | ERROR    |
decomp:get_tx_delay:246 - Packet 3730

phy.in_t or phy.in_t not present
2024-12-19 16:11:18.666 | ERROR    |
decomp:get_tx_delay:246 - Packet 3729

phy.in_t or phy.in_t not present
2024-12-19 16:11:18.667 | ERROR    |
decomp:get_tx_delay:246 - Packet 3729

phy.in_t or phy.in_t not present
2024-12-19 16:11:18.669 | ERROR    |
decomp:get_tx_delay:246 - Packet 3729

phy.in_t or phy.in_t not present
2024-12-19 16:11:18.671 | ERROR    |
decomp:get_tx_delay:246 - Packet 3729

phy.in_t or phy.in_t not present
2024-12-19 16:11:18.745 | ERROR    |
decomp:get_tx_delay:246 - Packet 142

phy.in_t or phy.in_t not present
2024-12-19 16:11:18.746 | ERROR    |
decomp:get_tx_delay:246 - Packet 142
phy.in_t or phy.in_t not present
```

Export csv

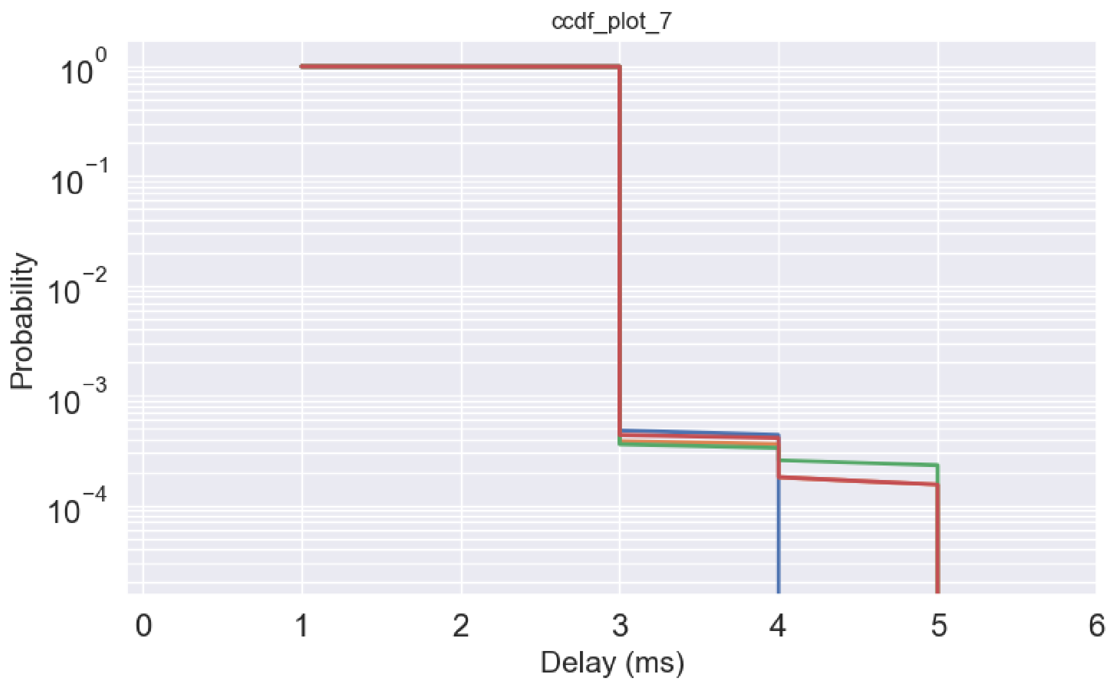
```
[70]: Meas_s40_s6263_s66.dataFrame(
          [
              "tbss",
              "segments",
              "segmentation_delays_wo_scheduling_delay",
          ],  # attrbutes name of Meas.delays
          [
              "TBS",
              "SegmentsNum",
              "SegmentDelay(noSched)",
          ],  # labels to display (optional)
      )
```

```
   TBS  SegmentsNum  SegmentDelay(noSched)
0   24            4              11.732101
1  116            1               2.104998
2   24            4              12.003183
3   24            3               9.509802
4   24            1               9.574890
./data/csv/
Dataframe saved to ./data/csv/s40_TBS_SegmentsNum_SegmentDelay(noSched).csv
   TBS  SegmentsNum  SegmentDelay(noSched)
0   24            3               9.410858
1   24            3               9.418011
2   24            4              11.901855
3   24            4              11.814833
4   24            3               9.358883
Dataframe saved to ./data/csv/s62_TBS_SegmentsNum_SegmentDelay(noSched).csv
   TBS  SegmentsNum  SegmentDelay(noSched)
0   24            1               9.531975
1   24            3               9.627819
2   24            1               9.535789
3   24            3               9.494066
4   24            2              11.957884
Dataframe saved to ./data/csv/s63_TBS_SegmentsNum_SegmentDelay(noSched).csv
   TBS  SegmentsNum  SegmentDelay(noSched)
0   24            1               9.522915
1   24            3               9.600878
2   24            2              12.004852
3   24            3               9.702682
4   24            1               9.557724
Dataframe saved to ./data/csv/s66_TBS_SegmentsNum_SegmentDelay(noSched).csv
```

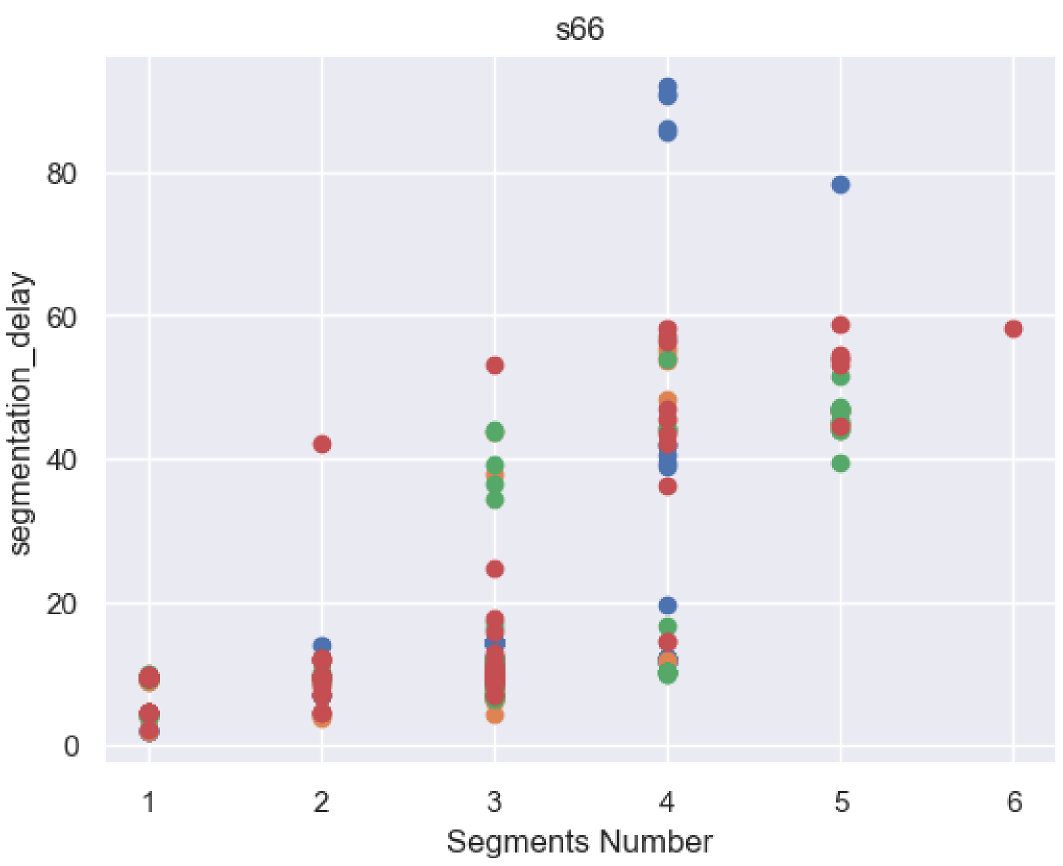
```
[71]: Meas_s40_s6263_s66.plotCCDF(
          ["segmentation_delays_wo_scheduling_delay"],
          ["s40", "s62", "s63", "s66"],
      )
```



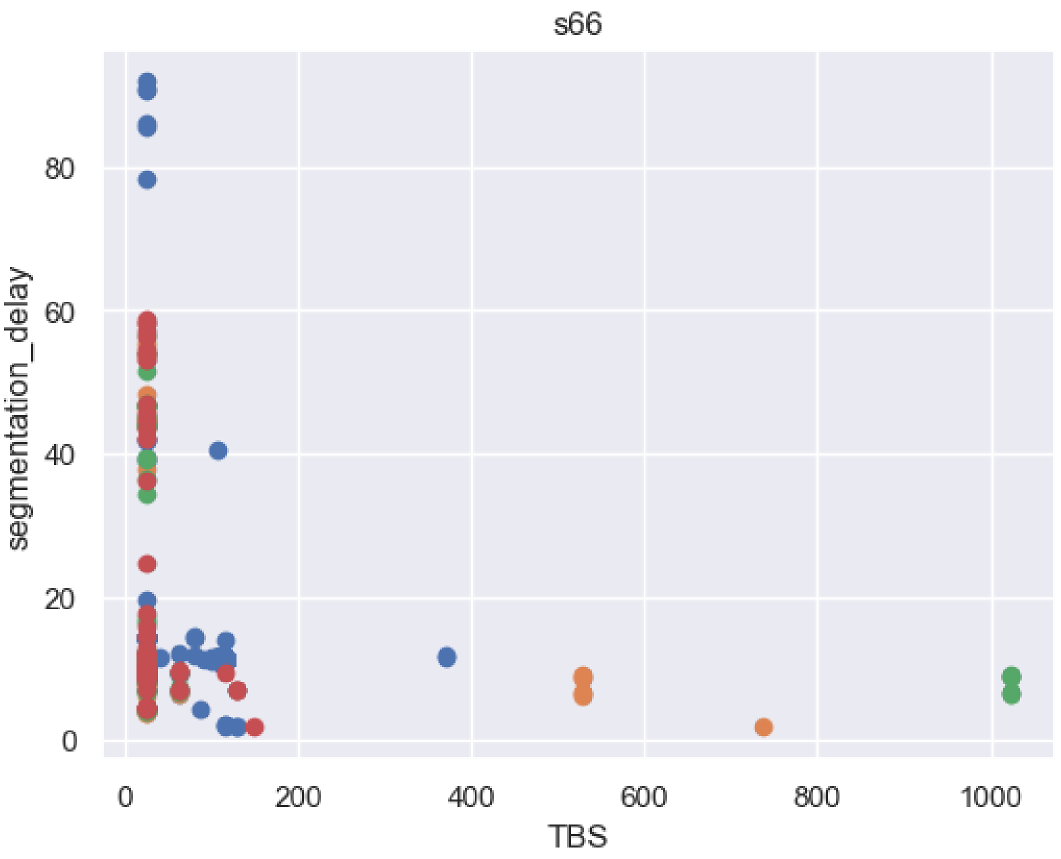
```
[72]: Meas_s40_s6263_s66.plotCCDF(
          ["segments"],
          ["s40", "s62", "s63", "s66"],
      )
```



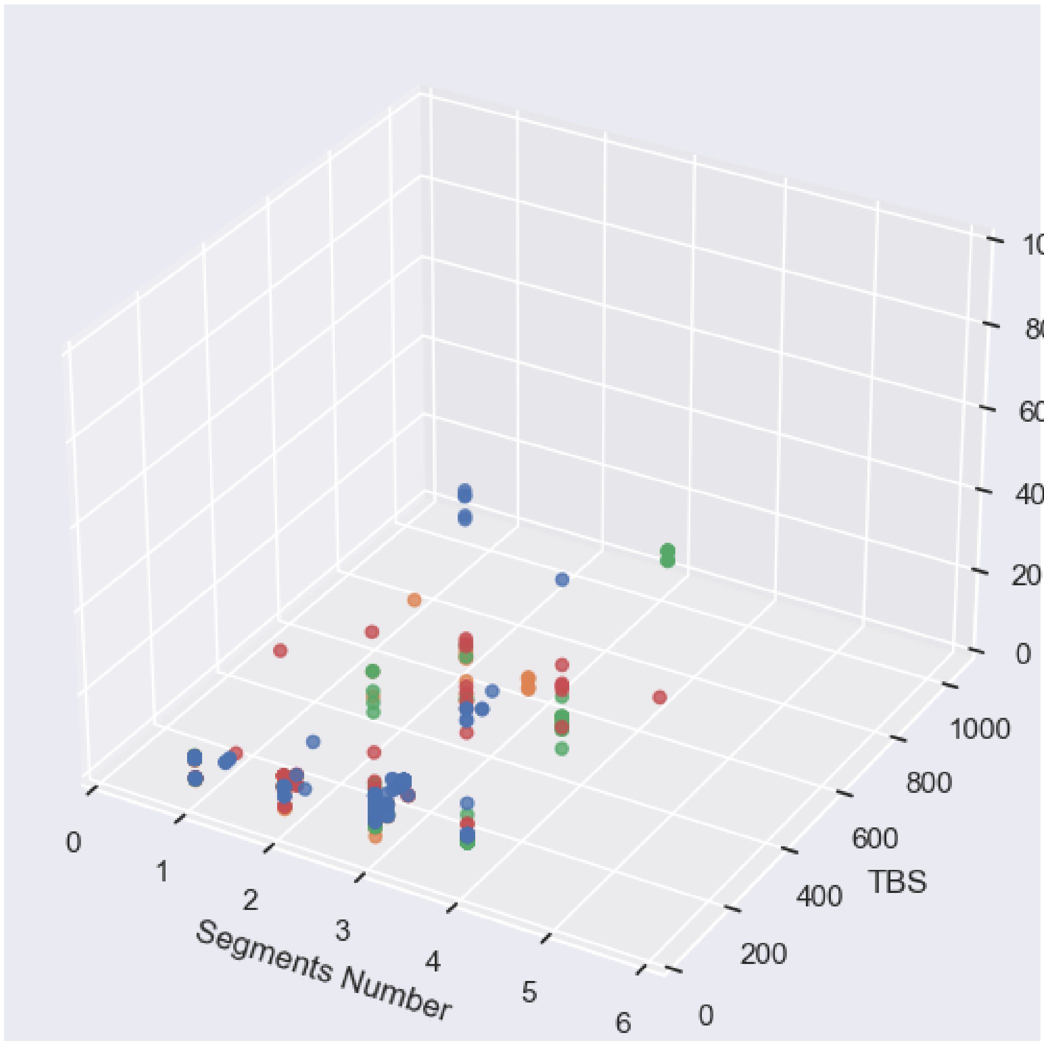
[73]:
```
for i in range(0,4):
    x = Meas_s40_s6263_s66.meas[i].delays.segments
    y = Meas_s40_s6263_s66.meas[i].delays.
 ↪segmentation_delays_wo_scheduling_delay
    plt.scatter(x,y)
    plt.xlabel("Segments Number")
    plt.ylabel("segmentation_delay")
    plt.title(f"{Meas_s40_s6263_s66.meas[i].meas_label}")
```



```
[74]: for i in range(0, 4):
          x = Meas_s40_s6263_s66.meas[i].delays.tbss
          y = Meas_s40_s6263_s66.meas[i].delays.
       ↪segmentation_delays_wo_scheduling_delay
          plt.scatter(x, y)
          plt.xlabel("TBS")
          plt.ylabel("segmentation_delay")
          plt.title(f"{Meas_s40_s6263_s66.meas[i].meas_label}")
```



[75]:
```python
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(7,7))
#     3D
ax = fig.add_subplot(projection="3d")
#
#c=["b","orange","g","r"]
for i in range(0, 4):
    x = Meas_s40_s6263_s66.meas[i].delays.segments
    y = Meas_s40_s6263_s66.meas[i].delays.tbss
    z = Meas_s40_s6263_s66.meas[i].delays.
 ↪segmentation_delays_wo_scheduling_delay
    #   3D
    ax.scatter(x, y, z, marker="o", alpha=0.8)
    #
#ax.view_init(elev=20, azim=290)
ax.set_xlabel("Segments Number")
ax.set_xlim([0, 6.1])
ax.set_ylabel("TBS")
ax.set_ylim([0,1080])
ax.set_zlabel("Segmentation Delay")
ax.set_zlim([0, 100])
#
# fig.tight_layout()
#plt.margins(2)
plt.show()
```


Segments Number
TBS