Three-way_timestamps_Rpi

October 6, 2019

1 Comparison of timestamps in three alternative NMEA data paths

1.0.1 Rasbian "Buster" - OpenCPN v5.0.0 - Signal K v1.17.0 - DashT v.0.5.2

We observe a five minute sampling period stored in InfluxDB database for each of the use case for single value of Apparent Wind Angle:

1. data via Signal K delta TCP channel with Signal K timestamps at its own reception
2. data via Signal K to NMEA-0183 via TCP channel timestamps at reception at the InfluxDB instruments
3. data directly from USB to OpenCPN

In all above cases the USB is set to 115200 baud at reception on Rasbian "Buster" (Raspberry Pi 4B 4GB) running OpenCPN v5.0.0. Data is originated from Raymarine SeaTalk (4800 baud) and converted to USB in MiniPlex II multiplexer - about 40 values per second are transmitted through this channel but only True Wind Angle timestamp behaviour is observed in this test.

[1]:
```
import numpy as np
import pandas as pd
```

1.1 1. Data via Signal K delta TCP channel

[2]:
```
df = pd.read_csv("2019-10-05_224308_SignalK_delta_zoom_Rpi.csv", sep=',',␣
↪header=3)
```

[3]:
```
df.head()
```

[3]:

	Unnamed: 0	result	table	_start	_stop	\
0	NaN	NaN	0	2019-10-05T16:14:00Z	2019-10-05T16:19:00Z	
1	NaN	NaN	0	2019-10-05T16:14:00Z	2019-10-05T16:19:00Z	
2	NaN	NaN	0	2019-10-05T16:14:00Z	2019-10-05T16:19:00Z	
3	NaN	NaN	0	2019-10-05T16:14:00Z	2019-10-05T16:19:00Z	
4	NaN	NaN	0	2019-10-05T16:14:00Z	2019-10-05T16:19:00Z	

	_time	_value	_field	_measurement	prop1
0	2019-10-05T16:14:01.169Z	-28.1	angleTrue	environment	wind
1	2019-10-05T16:14:03.106Z	-26.6	angleTrue	environment	wind

```
2  2019-10-05T16:14:05.045Z   -26.1  angleTrue  environment  wind
3  2019-10-05T16:14:06.049Z   -30.1  angleTrue  environment  wind
4  2019-10-05T16:14:07.956Z   -33.1  angleTrue  environment  wind
```

[4]:
```
df._value.describe()
```

[4]:
```
count    166.000000
mean     -21.151807
std       11.759992
min      -46.600000
25%      -28.975000
50%      -22.100000
75%      -14.725000
max       26.500000
Name: _value, dtype: float64
```

[5]:
```
df1 = pd.to_datetime(df['_time'])
```

[6]:
```
df1.describe()
```

[6]:
```
count                                166
unique                               166
top       2019-10-05 16:15:40.881000+00:00
freq                                   1
first     2019-10-05 16:14:01.169000+00:00
last      2019-10-05 16:18:59.339000+00:00
Name: _time, dtype: object
```

[7]:
```
df2 = df1.astype(np.int64).div(1e6)
```

[8]:
```
df3 = df2.diff()
```

[9]:
```
df3.describe()
```

[9]:
```
count     165.000000
mean     1807.090909
std       313.462833
min      1000.000000
25%      1863.000000
50%      1931.000000
75%      1942.999756
max      2225.000000
Name: _time, dtype: float64
```

1.2 2. Data via Signal K to NMEA-0183 converter TCP channel

[10]:
```
nf = pd.read_csv("2019-10-05_224509_SignalK_NMEA_TCP_zoom_Rpi.csv", sep=',',␣
↪header=3)
```

[11]:
```
nf.head()
```

[11]:
```
   Unnamed: 0  result  table                 _start                  _stop  \
0         NaN     NaN      0  2019-10-05T16:23:00Z  2019-10-05T16:28:00Z
1         NaN     NaN      0  2019-10-05T16:23:00Z  2019-10-05T16:28:00Z
2         NaN     NaN      0  2019-10-05T16:23:00Z  2019-10-05T16:28:00Z
3         NaN     NaN      0  2019-10-05T16:23:00Z  2019-10-05T16:28:00Z
4         NaN     NaN      0  2019-10-05T16:23:00Z  2019-10-05T16:28:00Z

                      _time  _value     _field _measurement prop1
0  2019-10-05T16:23:00.423Z    19.6  angleTrue  environment  wind
1  2019-10-05T16:23:02.379Z    26.6  angleTrue  environment  wind
2  2019-10-05T16:23:04.295Z    26.1  angleTrue  environment  wind
3   2019-10-05T16:23:05.38Z    26.1  angleTrue  environment  wind
4  2019-10-05T16:23:07.104Z    40.6  angleTrue  environment  wind
```

[12]:
```
nf._value.describe()
```

[12]:
```
count    188.000000
mean      21.622872
std        9.844945
min        0.500000
25%       13.850000
50%       22.850000
75%       29.600000
max       41.200000
Name: _value, dtype: float64
```

[13]:
```
nf1 = pd.to_datetime(nf['_time'])
```

[14]:
```
nf1.describe()
```

[14]:
```
count                                188
unique                               188
top       2019-10-05 16:24:50.701000+00:00
freq                                   1
first     2019-10-05 16:23:00.423000+00:00
last      2019-10-05 16:27:59.431000+00:00
Name: _time, dtype: object
```

[15]:
```
nf2 = nf1.astype(np.int64).div(1e6)
```

[16]:
```
nf3 = nf2.diff()
```

[17]:
```
nf3.describe()
```

[17]:
```
count     187.000000
mean     1598.973262
std       422.940687
min      1000.000000
25%      1026.500000
50%      1877.000000
75%      1928.000000
max      2036.000000
Name: _time, dtype: float64
```

1.3 3. Data without Signal K directly from USB

[18]:
```
of = pd.read_csv("2019-10-05_224657_USB_to_0_zoom_Rpi.csv", sep=',', header=3)
```

[19]:
```
of.head()
```

[19]:

	Unnamed: 0	result	table	_start	_stop	\
0	NaN	NaN	0	2019-10-05T16:34:00Z	2019-10-05T16:39:00Z	
1	NaN	NaN	0	2019-10-05T16:34:00Z	2019-10-05T16:39:00Z	
2	NaN	NaN	0	2019-10-05T16:34:00Z	2019-10-05T16:39:00Z	
3	NaN	NaN	0	2019-10-05T16:34:00Z	2019-10-05T16:39:00Z	
4	NaN	NaN	0	2019-10-05T16:34:00Z	2019-10-05T16:39:00Z	

	_time	_value	_field	_measurement	prop1
0	2019-10-05T16:34:01.719Z	8.1	angleTrue	environment	wind
1	2019-10-05T16:34:03.658Z	9.1	angleTrue	environment	wind
2	2019-10-05T16:34:05.585Z	4.1	angleTrue	environment	wind
3	2019-10-05T16:34:07.545Z	1.6	angleTrue	environment	wind
4	2019-10-05T16:34:09.462Z	7.6	angleTrue	environment	wind

[20]:
```
of._value.describe()
```

[20]:
```
count    172.000000
mean      16.232558
std       10.655020
min        0.500000
25%        7.900000
50%       15.600000
75%       22.225000
max       52.100000
Name: _value, dtype: float64
```

[21]:
```
of1 = pd.to_datetime(of['_time'])
```

[22]:
```
of1.describe()
```

[22]:
```
count                                  172
unique                                 172
top       2019-10-05 16:36:01.753000+00:00
freq                                     1
first     2019-10-05 16:34:01.719000+00:00
last      2019-10-05 16:38:58.943000+00:00
Name: _time, dtype: object
```

[23]:
```
of2 = of1.astype(np.int64).div(1e6)
```

[24]:
```
of3 = of2.diff()
```

[25]:
```
of3.describe()
```

[25]:
```
count     171.000000
mean     1738.152047
std       366.632300
min      1000.000000
25%      1825.499634
50%      1925.000000
75%      1942.500000
max      2025.999756
Name: _time, dtype: float64
```

1.3.1 Applying moving average

[26]:
```
af = pd.read_csv("2019-10-05_224657_USB_to_0_zoom_Rpi_moving_average.csv",␣
↪sep=',', header=3)
```

[27]:
```
af.head()
```

[27]:

	Unnamed: 0	result	table	_start	_stop	_time	_value	_field	_measurement	prop1
0	NaN	NaN	0	2019-10-05T16:34:00Z	2019-10-05T16:39:00Z	2019-10-05T16:34:33.669Z	18.570	angleTrue	environment	wind
1	NaN	NaN	0	2019-10-05T16:34:00Z	2019-10-05T16:39:00Z	2019-10-05T16:34:35.605Z	18.795	angleTrue	environment	wind
2	NaN	NaN	0	2019-10-05T16:34:00Z	2019-10-05T16:39:00Z	2019-10-05T16:34:37.541Z	18.745	angleTrue	environment	wind
3	NaN	NaN	0	2019-10-05T16:34:00Z	2019-10-05T16:39:00Z	2019-10-05T16:34:39.468Z	18.995	angleTrue	environment	wind
4	NaN	NaN	0	2019-10-05T16:34:00Z	2019-10-05T16:39:00Z	2019-10-05T16:34:41.444Z	19.365	angleTrue	environment	wind

[28]:
```
af._value.describe()
```

[28]:
```
count    153.000000
mean      16.781928
std        5.240853
min        5.600000
25%       13.300000
50%       17.200000
75%       20.085000
max       27.700000
Name: _value, dtype: float64
```

1.4 Summary of results

data path	timestamp	standard deviation	maximum time difference
1 Signal K delta	at source	313 ms	2225 ms
2 Signal K NMEA TCP	at reception	423 ms	2036 ms
3 USB to OpenCPN	at reception	367 ms	2025 ms

1.5 Conclusion

Judged by a human eye there is no difference between the three methods - the needles and values are jumping back and forth as always!

The difference will come apparent when we want to eliminate that jumping by applying some statistical and continuous algorithms on the received time series data. The accuracy of the time stamps is, of course important for any time series analysis.

1. It is not surprising that the direct TCP connection to the Signal K emitted delta values is the most efficient what comes to the accuracy of the timestamps - they are set at the reception, *i.e.* at the closest possible position to the source. Although this method is penalized having to transmit also information in its payload to which we are not necessarily willing to be subscribed, the fact that the timestamp travels with the data compensates that inconvenience.

2. The fact that there is so little difference between the timestamp accuracy through the Signal K to NMEA conversion and its actual delta channel is a proof of the excellent quality and effiency of Signal K and npm. Also, the TCP method of OpenCPN is the preferred one since apparently well implemented.

3. One observation in this test using a modest Rasberry Pi 4B board is that it allows similar performances than more powerful i7 CPU based processor under Window 10. Also, the USB streaming implementationin wxWidgets/OpenCPN is now in equal or even better performance than the combined Signal K (USB input) with conversion to TCP which OpenCPN receives. However, the difference in jitter is not significative enough to justify a need to elimate Signal K from the signal chain: using it together with method (1) - direct delta connection - provides still the best overall performance for key parameters.

Finally, the best improvement in this particular case would be to increase the sampling rate, which is, admittely, ridicuosly slow.