THE EDGE COMPUTING ALGORITHM AND NETWORK PERFORMANCE ANALYSIS

1. Introduction

A monitoring scheme testbed is developed to monitor a model of a power network integrated with a wind enregy DER simulated on the RTDS. Data from the model is acquired by an EC gateway model over a LAN connection. The gateway model consists of RTAC SEL-3555 and two IEC 61850 communication drivers in Elipse Power application. The RTAC SEL-3555 forwards data over a local area connection to the first driver in Elipse Power which is an IEC 61850-8-1 MMS client. Thereafter, data is forwarded to the second driver which is an IEC 61850-8-2 XMPP server that enables XMPP-based communication over WANs.

The workstation ‘PC-3’ which represents a control center in a distant location, hosts a remote IEC 61850-8-2 XMPP client driver that polls data from the IEC 61850-8-2 XMPP server driver on ‘PC-1’. The XMPP-based communications are managed by the Openfire XMPP server application implemented on ‘PC-2’. The chapter presents an evaluation of the EC impact on communications QoS for monitoring remote DER sites in a smart grid. An algorithm for data fusion is implemented in RTAC SEL-3555 to reduce data volume before transmission over an Internet-based WAN. Wireshark software is used to capture and analyze the network traffic to measure the QoS metrics of interest, which are latency and bandwidth usage. Additionally, data packets in the network traffic are analyzed to examine IEC 61850-8-2 XMPP standard data exchange and cybersecurity mechanisms.

2. Setting up Wireshark

Wireshark is a prominent open-source software for capturing and analysing data traffic in communication networks. The host machine of Wireshark was plugged into the main switch of the lab-scale monitoring testbed as described in figure 1 below. It was operated to capture IEC 61850-8-2 XMPP communications traffic over an Internet-based WAN.

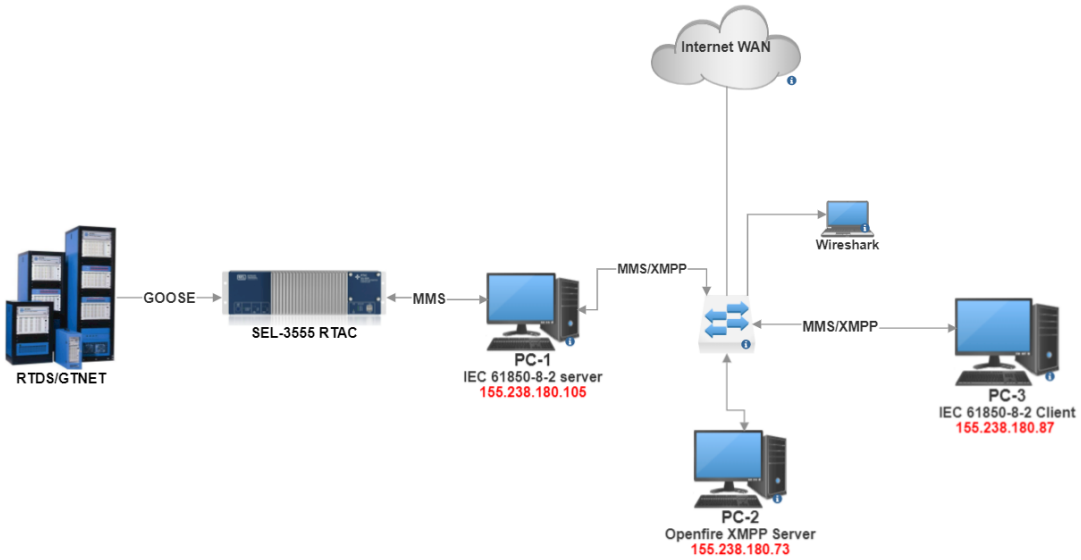

Figure 1: **Wireshark capturing point in the lab-scale testbed.**

The utilized network switch is configured to reroute traffic from all ports to a monitoring port, where Wireshark machine is plugged. This was achieved by activating the 'Mirror' port feature. The 'Mirror' port allows Wireshark to monitor all network traffic between XMPP clients and Openfire server. This point is the best option to get an accurate and synchronized capture of network traffic.

The following section presents a visual break down of the XMPP data exchange and cybersecurity aspects using the captured packets.

3. Analysis of IEC 61850-8-2 XMPP standard data exchange and cybersecurity mechanisms

The section describes a visual analysis of the data exchange and cybersecurity mechanisms of IEC 61850-8-2 XMPP standard. The data exchange profile specifies that IEC 61850 devices are to be hosted by XMPP clients. This is implemented in Elipse Power IEC 61850 drivers as they host the IEC 61850-8-1 MMS client/server entities, and uses XMPP as the transport layer protocol. All XMPP clients are linked across a WAN to an XMPP server and are given unique JID addresses (Nadeem et al., 2019). The Client/server architecture is used in XMPP communications whereby TCP/IP connections to the XMPP server are established by XMPP clients. The data exchange stream captured by Wireshark, between IEC 61850-8-2 XMPP client in 'PC-1' with IP address "155.238.180.105" and Openfire XMPP server hosted in 'PC-2' with IP address "155.238.180.73", are illustrated in figure 2 below.

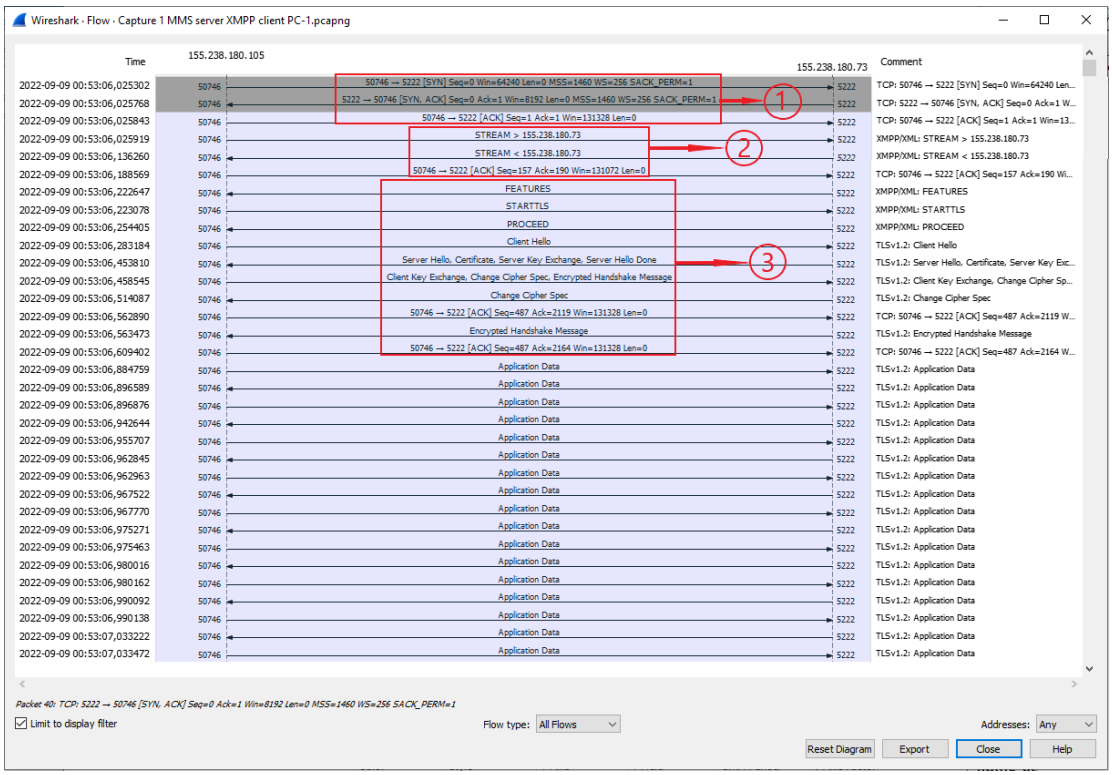

Figure 2: XMPP exchanged packets between client/server.

The XMPP client initiates communications by sending a connection request, and the server responds to establish a TCP/IP connection as seen in the first three packets highlighted with number one. A magnified view of the exchanged data packets, is provided in figure 3 below. Once a TCP/IP link is established, an XML stream is enabled between client/server as indicated in the second packet group. Thereafter, a TLS connection is negotiated between client/server as highlighted in the third packet group.

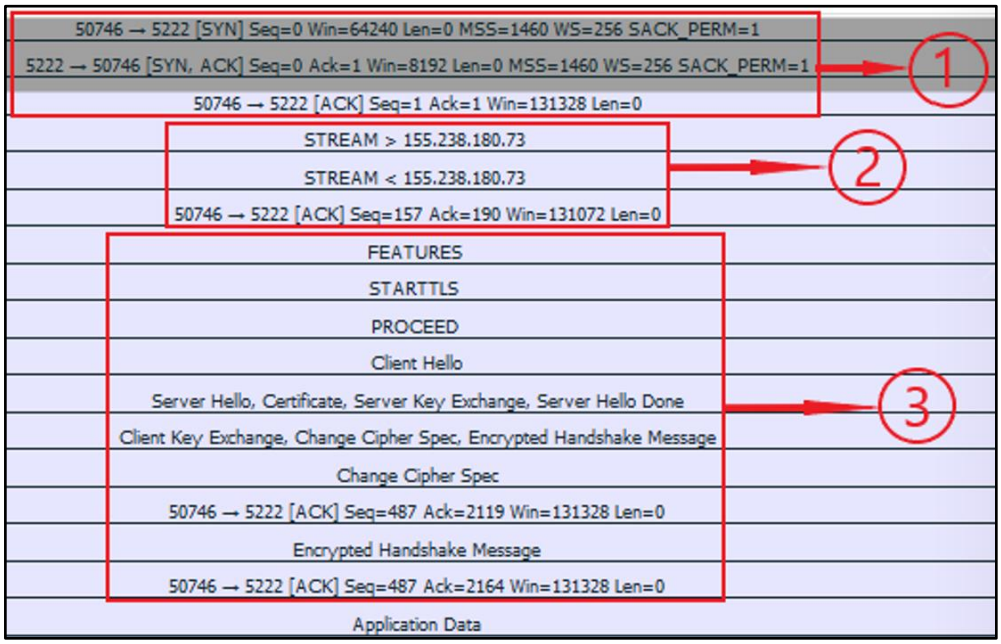

Figure 3: A magnified view of the XMPP exchanged packets.

The XMPP-based communications take place in secured and an encrypted fashion. The TLS and SASL security protocols are implemented at the transport layer in E2M XMPP communication. The TLS protocol provides channel encryption to secure data from tampering and eavesdropping (Saint-Andre, 2011). As noted in TLS negotiation packets in figure 3 above, the client receives the 'Features' packet from the server and then initiates a 'STARTTLS' command.

A handshake is intitatied by the client 'Client Hello' and the server responds by sending a security certificate and a server key "authentication identity". The client negotiates by requesting a 'Change Cipher Spec' to the server which is a request to change the encryption cipher specifications to be applied. The server acknowledges this request of change, and a secured and encrypted transport channel will be established, which will lead to realizing integrity and confidentiality for E2M XMPP communications. Consequently, upon completion of a successful TLS negotiation, the E2E SecProtocol authentication messages are interchanged in the form of encrypted handshakes to verify the IEC 61850 MMS client/server end peers as authenticated users.

The 'Features' packet is the first to be sent by an XMPP server to any XMPP client after establishing the XML stream. It carries all the server's information in the XMPP payload. As depicted in figure 4 below, the XMPP server declares all its features to the client including server information, cybersecurity mechanisms, encryption cipher, and compression method. After receiving this packet, the client requests to start TLS negotiations.

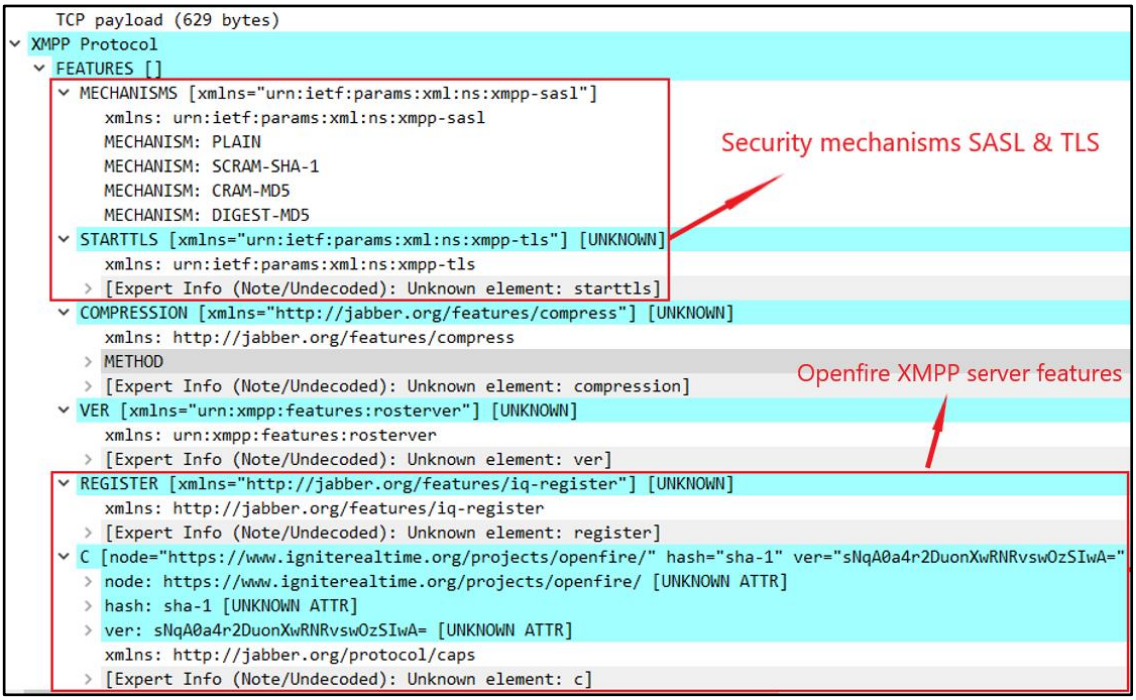

Figure 4: XMPP server 'Features' packet contents.

As seen in figure 5 below, the contents of an ‘Application Data’ packet are encrypted and any attempt from possible attackers to eavesdrop on XMPP communications using packet-capturing tools such as Wireshark, would be useless to get any information.

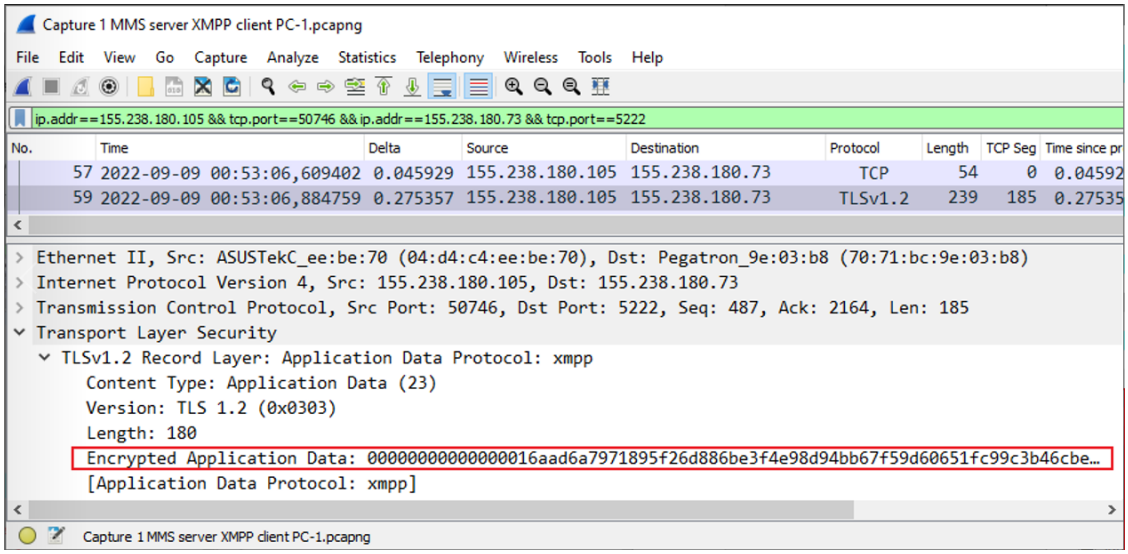

Figure 5: **The ‘Application Data’ packet encrypted content.**

The following section presents the developed algorithm to reduce the data volume transmitted on a network.

4. Edge computing algorithm for data reduction

A data fusion algorithm is implemented in RTAC SEL-3555 using IEC 61131-3 ST language. The main purpose of the algorithm is to reduce the number of transmitted data tags over a network. In the research testbed, a total of twelve data tags are used for monitoring the power network model in RTDS listed in table 1 below.

Table 1: **Signal names of the data tags under the monitoring scheme**

Parameter	Value
Generator’s busbar phase A voltage	STLWT1a
Generator’s busbar phase B voltage	STLWT1b
Generator’s busbar phase C voltage	STLWT1c
Wind Gusts value	GUST
Computed Air Density	airdensity
Wind speed in Km/hr	windkph
Hub Speed in rad/sec	HUBSPD
Turbine Power in Mega Watts	pwrturb
Blades Pitch degree	pitchdeg

Rotor Speed in pu	STLWT1SPD
Stator active power (P) in Mega Watts	STLWT1P
Stator reactive power (Q) in Mega VAR	STLWT1Q

The first three points are root mean square (rms) values of single-phase voltages of the generator busbar 'STWLT1a, STWLT1b, STWLT1c'. The equivalent three-phase value can be transmitted instead of three individual values, it is calculated by using equation 1 below, where $\boldsymbol{V_A}$ represents the single-phase voltage.

$$\boldsymbol{V_{3ph}} = \sqrt{\boldsymbol{3}} * \boldsymbol{V_A} \quad (1)$$

The three tags, wind gust 'GUST', wind speed 'windkph', and pitch degree 'pitchdeg', are reduced to a single point by using the gust value as an alarm for wind overspeed which can damage the turbine. In the occurrence of wind gust, the gateway must transmit the pitch degree value to inform the control centre that, the pitch angle control is operative. In the event that, pitch angle control is not operative, a manual intervention would be needed to prevent turbine damage during high wind speeds. However, the gateway keeps monitoring the wind speed during normal and low wind speeds. This logic can be expressed in the following steps:

- If 'GUST' > 1
- Transmit pitch degree 'pitchdeg'
- Else
- Transmit wind speed 'windkph'

A similar concept is applied to combine the power output of the induction machine 'STLWT1P' and the power output of the wind turbine 'pwturb'. For the duration of the wind turbine startup and before reaching rated speed and coupling to the grid, the gateway must transmit the power output of the turbine. During this period, the power output of the generator would be very small or even a negative value. Thereafter, when rated speed is achieved and the generator is coupled to the grid, priority shifts to the generator power output to be monitored. This logic can be articulated in the following steps:

- If generator output power 'STLWT1P' < 0
- Transmit turbine output power 'pwturb'
- Else
- Transmit generator's output power 'STLWT1P'

The final part of the algorithm combines generator rotor speed ‘STLWT1SPD’ and turbine hub speed ‘HUBSPD’, which are interrelated via the gear ratio. Both values are calculated by the turbine model in RTDS, using different units. The rotor speed is provided in pu and the hub speed in radiance per second (rad/s). The rotor speed can be converted to turbine hub speed using the gear ratio value. Henceforth, the EC gateway can transmit only the difference between the converted and the monitored hub speed ‘HUBSPD’ instead of transmitting two data tags. The difference should be very small “optimally zero” and any significant change in the value would indicate existence of mechanical faults in the coupling parts between turbine and generator.

According to the technical specifications of the turbine model “Vestas V82” listed in table A.1 in the appendix, the gear ratio is “84.5”, and the rated speed of the generator machine is “1200” revolution per minute (rpm) which is the base value for the rotor speed in pu. In order to convert the rotor speed to hub speed, the following steps are performed using equations 2 up to 5:

- Convert pu to rpm

$$1\,pu \;=\; 1200\,rpm \tag{2}$$

$$STLWT1SPD \;*\; 1200 \;=\; rotor\ speed\ in\ rpm \tag{3}$$

- Apply the gear ratio to convert to the hub speed

$$rotor\ speed\ in\ rpm \;*\; 1/84.5 \;=\; hub\ speed\ in\ rpm \tag{4}$$

- Convert hub speed to rad/sec

$$hub\ speed\ in\ rad/s \;=\; hub\ speed\ in\ rpm \;*\; (2\pi/60) \tag{5}$$

The converted value of the hub speed in equation 5 above, is subtracted from the monitored hub speed value which is provided via GOOSE from the turbine model in RTDS, and the difference is transmitted by the gateway. The algorithm is developed using IEC 61131-3 ST language in RTAC SEL-3555 as shown in figure 6 below.

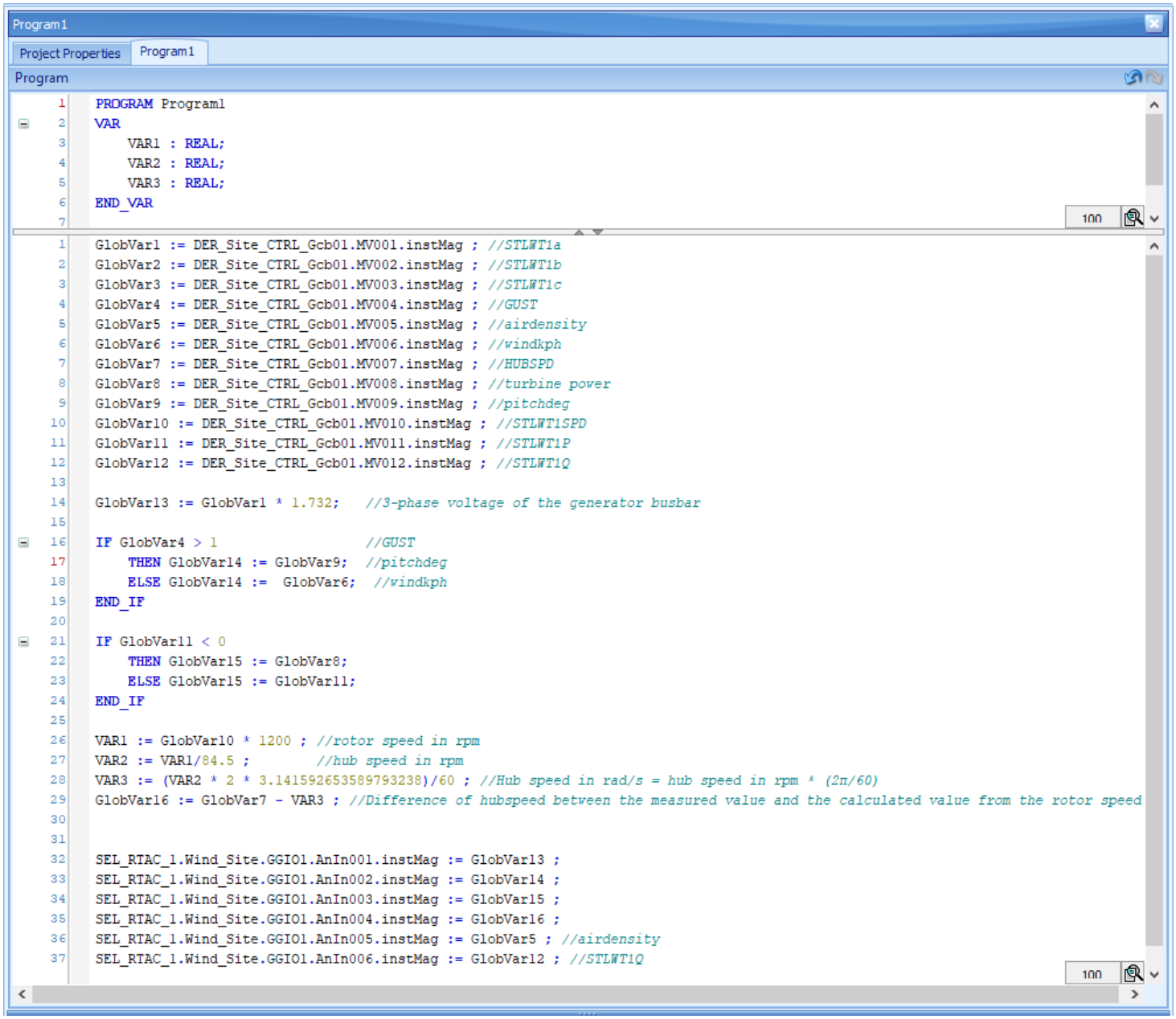

Figure 6: **Data fusion algorithm in RTAC SEL-3555.**

Simulations were run for the two test cases, with and without executing the data fusion algorithm. When the algorithm is not executed, all twelve data tags are transmitted without any reduction as displayed in figure 6 below. Data was tracked across the testbed in order to verify the algorithm effect. GOOSE messages which are published by the DER model on RTDS, were monitored using GOOSE Inspector software indicated by number one in the figure. Then, the GOOSE is mapped to the server model of RTAC SEL-3555, and the online monitoring mode in RTAC SEL-3555 was used to observe data values in real-time, as highlighted by number two in the figure. Finally, the online view of the IEC 61850-8-1 MMS client driver in Elipse Power was captured as indicated by number three in the figure.

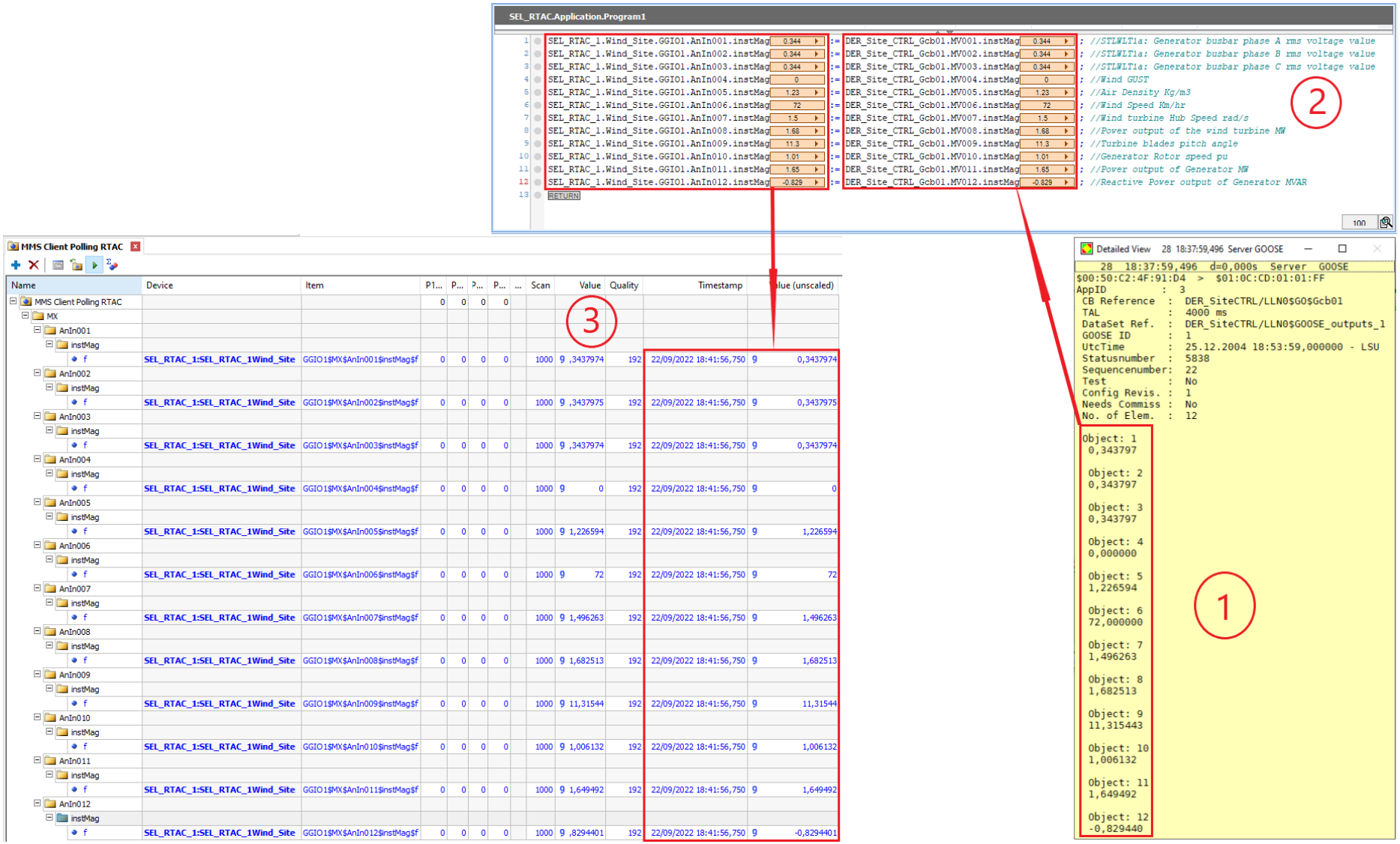

Figure 7: Data across the gateway model without execution of the EC algorithm.

After the data fusion algorithm was executed, the data tags were effectively reduced from twelve to six tags. As illustrated in figure 8 below, twelve data tags are received via GOOSE, assigned to global variables, then the algorithm operates on these variables. Thereafter, the resulting data tags are assigned to DOs of the RTAC SEL-3555's server model. Six tags are forwarded to the IEC 61850-8-1 MMS client driver in Elipse Power.

The following section presents an analysis of the communication performance using Wireshark packet-capturing software.

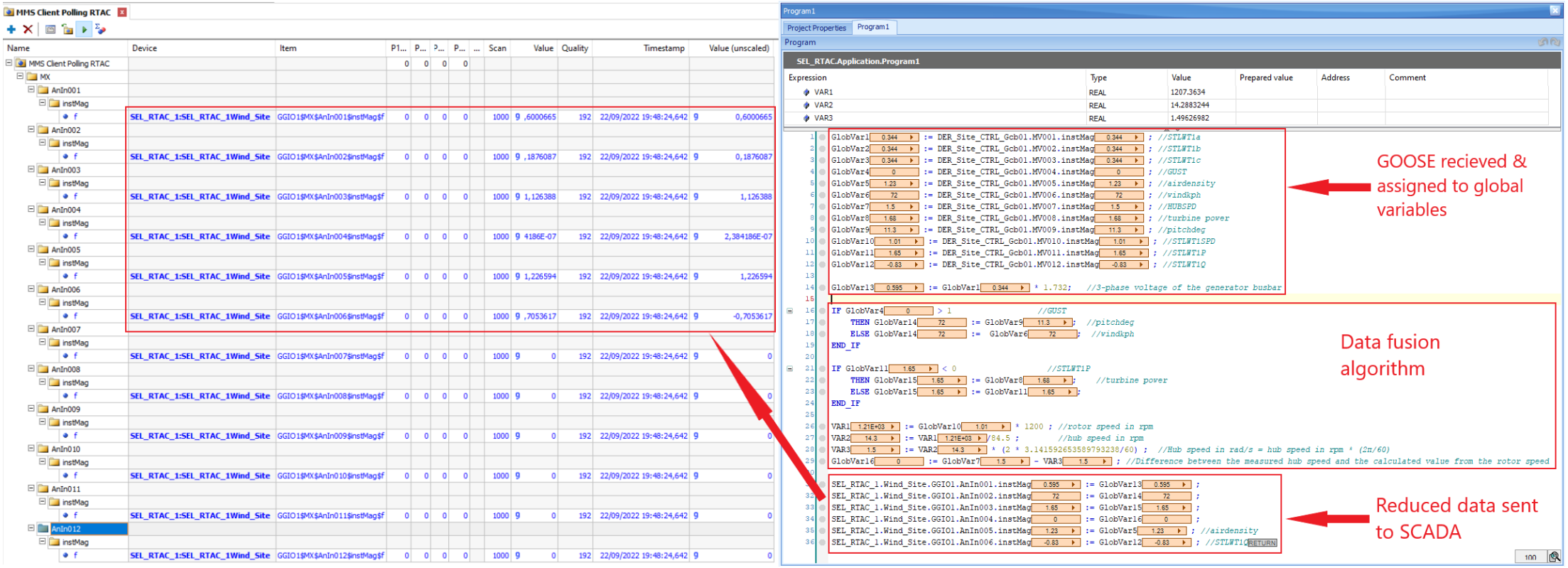

Figure 8: Data across the gateway model with the execution of the EC algorithm.

5. Analysis of the communication Quality of Service using Wireshark

The QoS metrics such as bandwidth and latency are performance indicators of communication networks. Bandwidth is the transmission capacity of a network that determines the data volume which can be transferred across a network in a specific period of time. Bandwidth is measured in bits/s, and is the main characteristic that decides the quality of a communication network. Insufficient bandwidth affects the overall network performance and causes congestions that result in excessive latencies, packet losses, and poor connectivity. Therefore, minimizing the bandwidth usage in a network is critical to ensure the best possible performance.

A major purpose of implementing the EC concept, is to reduce data traffic in a network by processing data near the source before transmission, consecuently, reducing the bandwidth usage will improve the overall network performance. In the testbed, Wireshark was employed to capture data traffic in the network and to provide measurements for communications QoS metrics “bandwidth usage and latency”. These measurements are used to evaluate the impact of implementing the EC algorithm on the performance of a communication network. The methodology of evaluation is established on comparing QoS measurements in two test cases, with and without executing the EC algorithm. The following sections present methodologies and results of Wireshark analysis tools.

5.1 Description of Wireshark ‘Capture Files’

The traffic captures for both test cases were taken for equal period of time twenty five minutes, to provide an accurate comparison of the network performance. A display filter was applied to extract the traffic of interest between the Openfire XMPP server and Elipse Power drivers “XMPP clients” in the testbed. The applied filter is written in the format “ip.addr == 155.238.180.73 && ip.addr == 155.238.180.105 or ip.addr == 155.238.180.73 && ip.addr == 155.238.180.87”. As a result, the displayed packets are only the conversations between the Openfire XMPP server hosted on ‘PC-2’ with IP address “155.238.180.73”, and the IEC 61850-8-2 XMPP clients hosted on ‘PC-1’ and ‘PC-3’ with IP addresses “155.238.180.105” and “155.238.180.87” respectively. Wireshark produces statistics for both captured and filtered data. The properties of the ‘Capture Files’ for the test cases are displayed in figures 9 and 10 below.

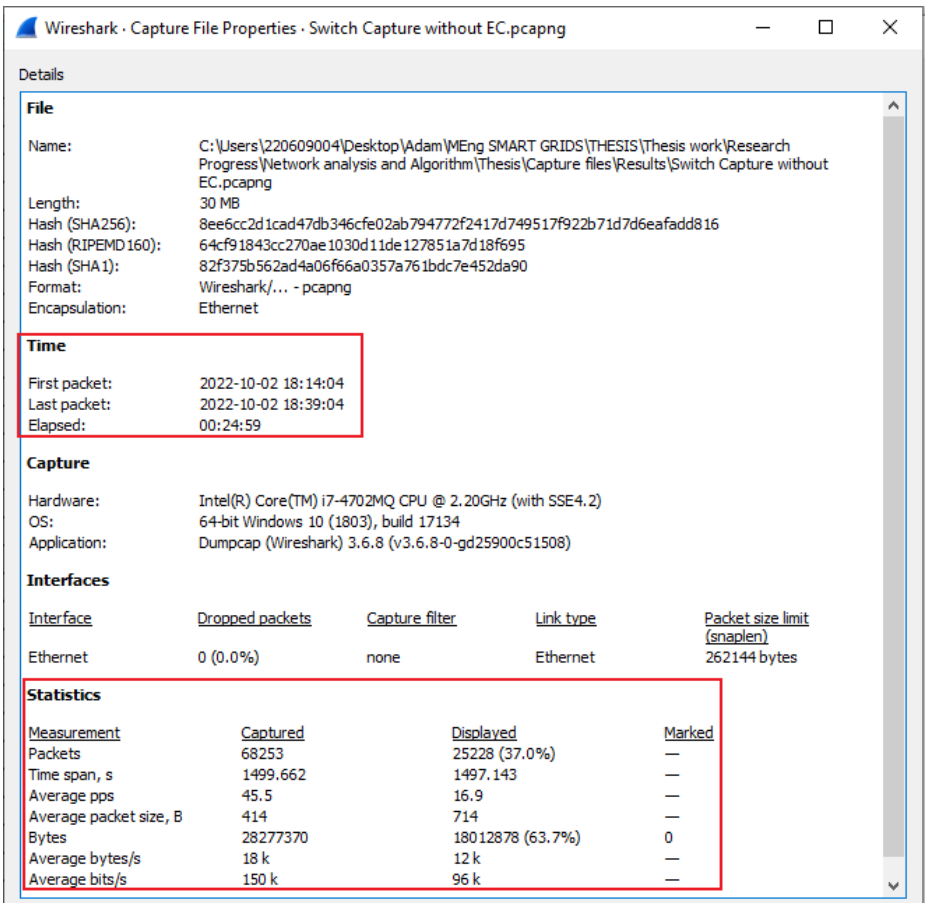

Figure 9: The capture file properties without execution of the EC algorithm.

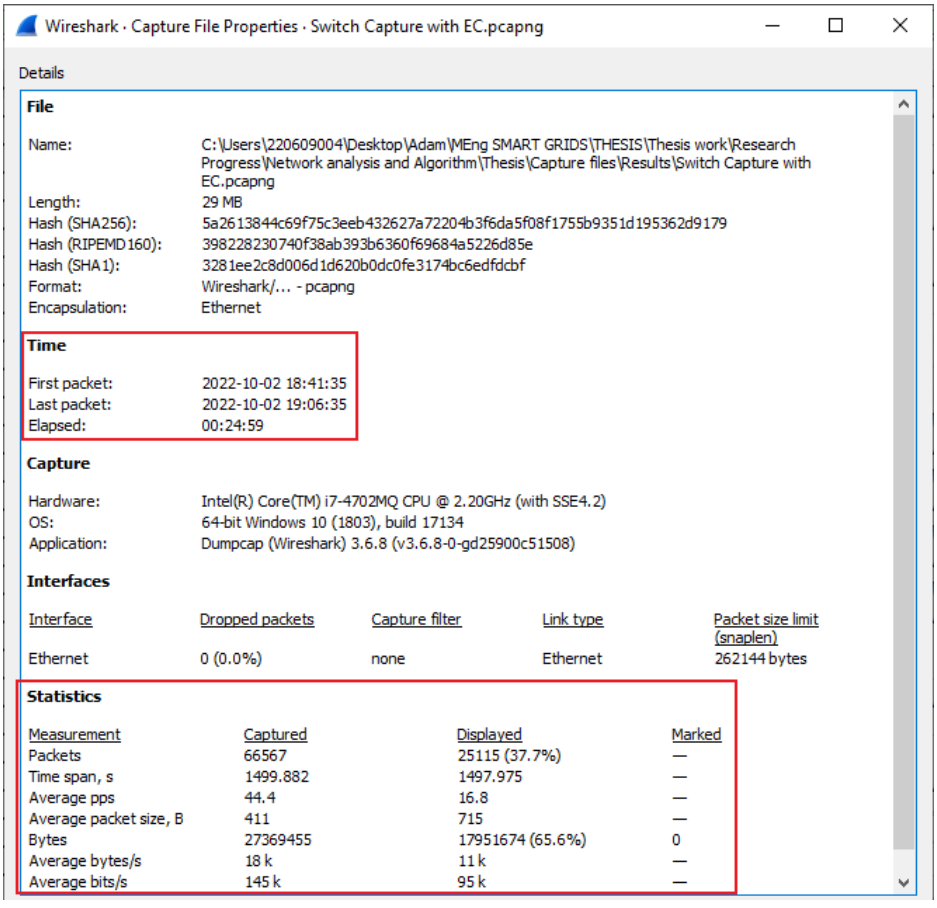

Figure 10: The capture file properties with the execution of the EC algorithm.

The displayed properties include the capture file name, length, date and time of the capture, and statistics of captured and displayed traffic “filtered XMPP data traffic”. As noticed from the ‘Statistics’ part in the figures above, the average bandwidth usage reflects the impact of the EC algorithm. In the capture file ‘Switch Capture without EC’ in figure 9 above, the bandwidth usage is ‘96 Kbits/s’ for the filtered XMPP traffic. While in the capture file ‘Switch Capture with EC’ in figure 10 above, the bandwidth usage reads ‘95 Kbits/s’. This indicates a bandwidth usage reduction by one Kbits/s.

Wireshark provides comprehensive tools to evaluate the bandwidth usage and latency from the captured traffic. The I/O Graphs tool gives accurate measurments of QoS metrics. The following section provides an analysis of communication QoS for both test cases in the testbed using Wireshark I/O Graphs.

5.2 Communication Quality of Service analysis using Wireshark I/O Graphs

The bandwidth usage in each test case of the implementation, are provided in figures 11 and 12 below.

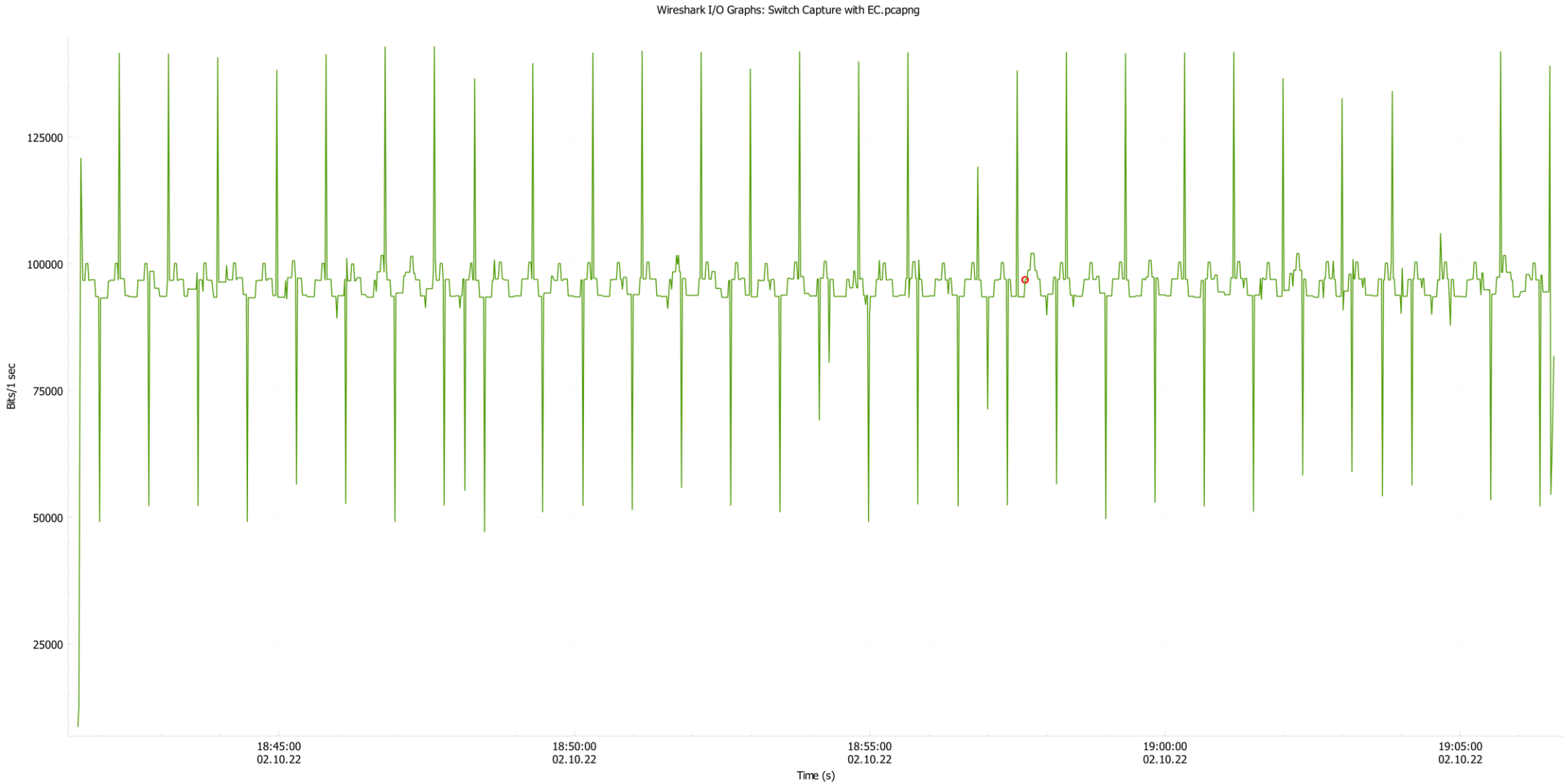

Figure 11: Bandwidth usage of XMPP communications with execution of the EC algorithm.

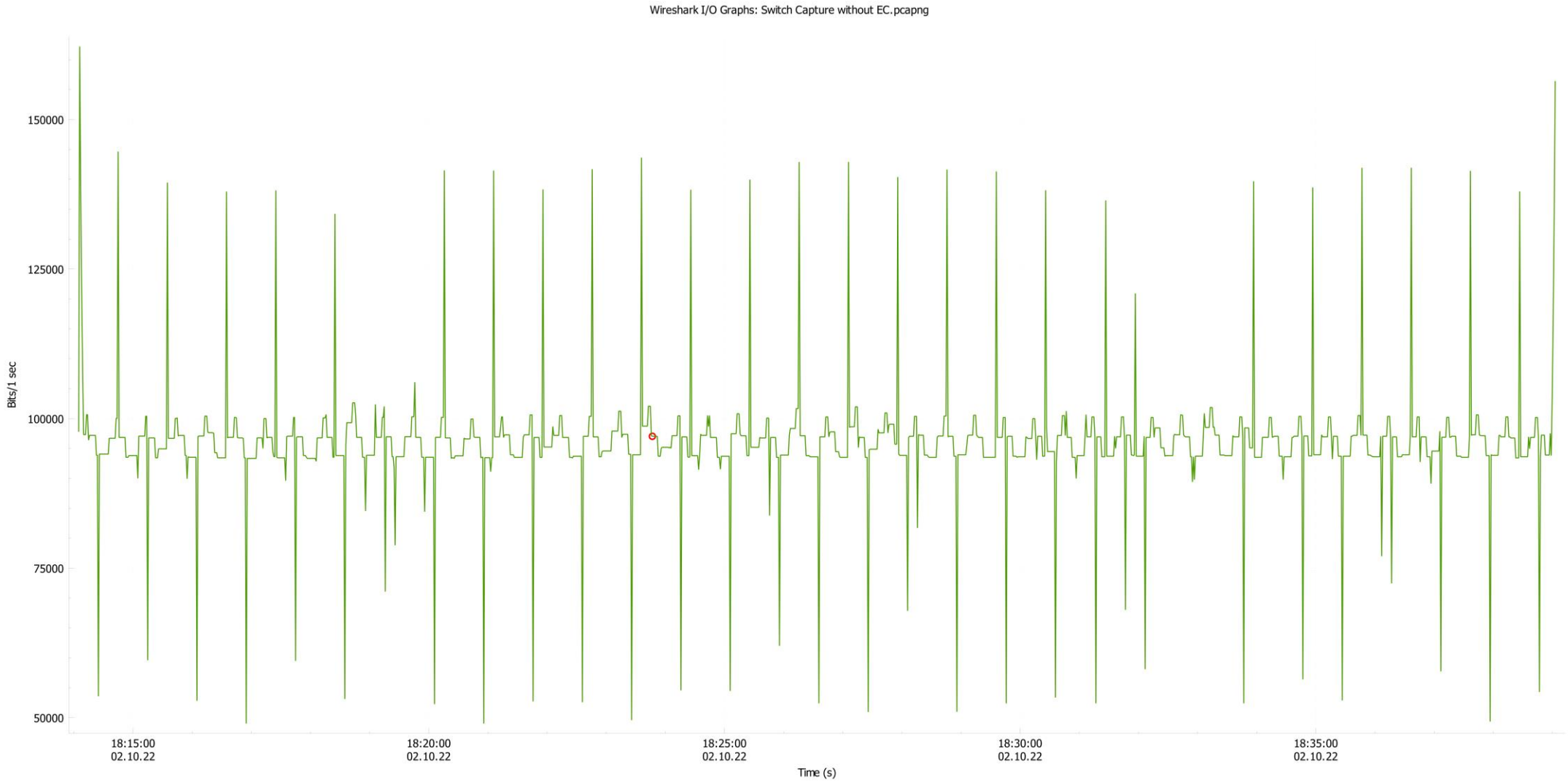

Figure 12: **Bandwidth usage of XMPP communications without execution of the EC algorithm.**

The latency is calculated from the time delay between consecutive packets in a TCP stream, which can be displayed using the filter expression “tcp.time-delta”. The I/O Graphs tool computes and plots the average latency values for each case as shown in figures 13 and 14 below.

A key feature of I/O Graphs is that, it allows for exporting the plots to a Comma Separated Values (CSV) format which can be further analysed using Microsoft Excel. Consequently, Microsoft Excel was used to calculate the average “arithmetic mean” of data values for bandwidth usage, and to determine the maximum latency recorded in each case. The bandwidth and latency results for each case are listed in table 2 below.

Table 2: The QoS results for the two test cases.

QoS parameter	With EC algorithm	Without EC algorithm	Impact
Average Bandwidth (bits/s)	95742.26	96132.77	390.51
Maximum Delay (s)	8.09644	8.53397	0.43753

From the results in table 2 above, the impact of the EC algorithm on the communication QoS is detected. The following section presents points of discussion on the testbed simulations, the recorded results, and the impact of EC on communication QoS.

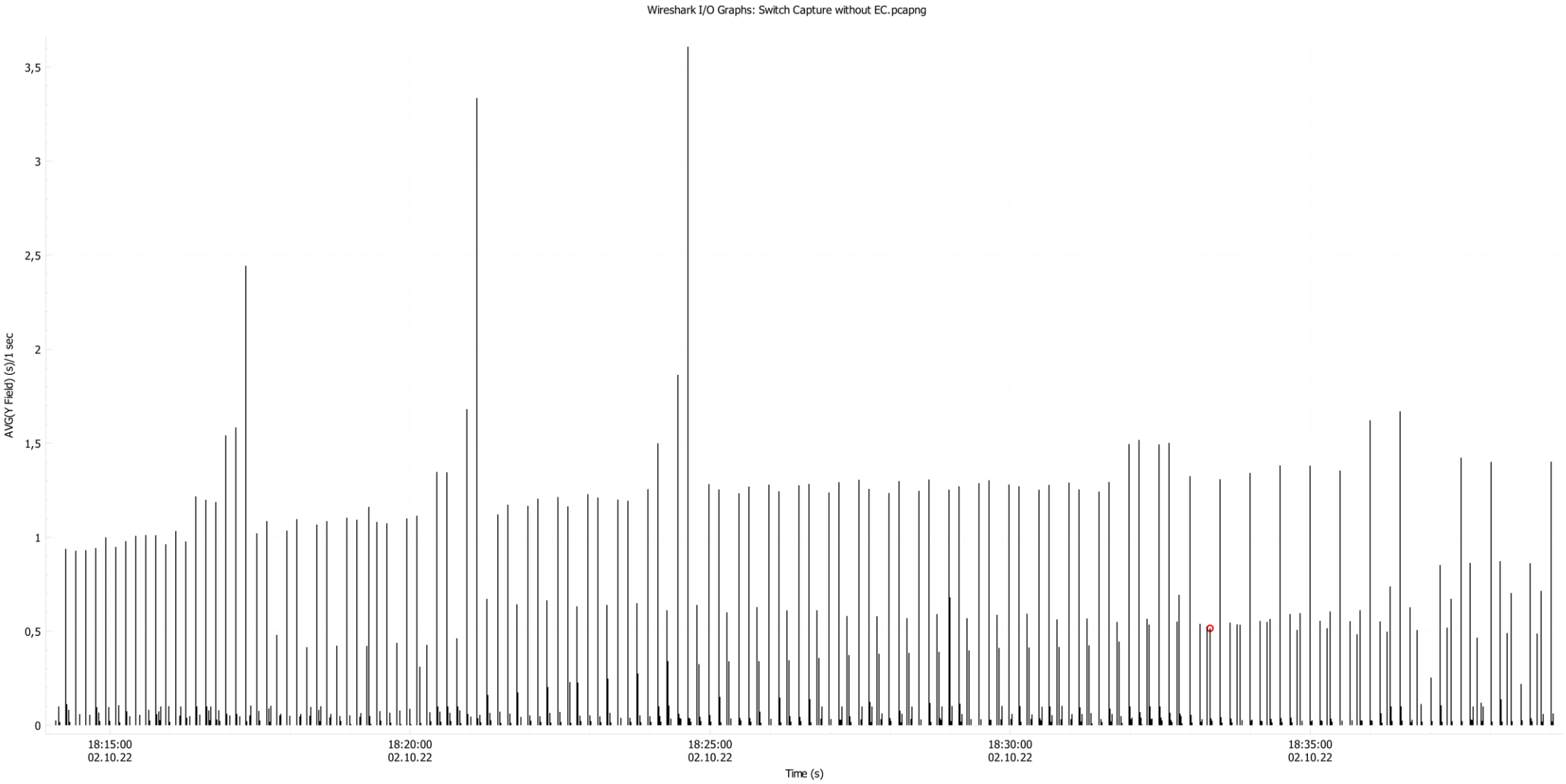

Figure 13: Average latencies in the TCP stream of the network traffic without the EC algorithm.

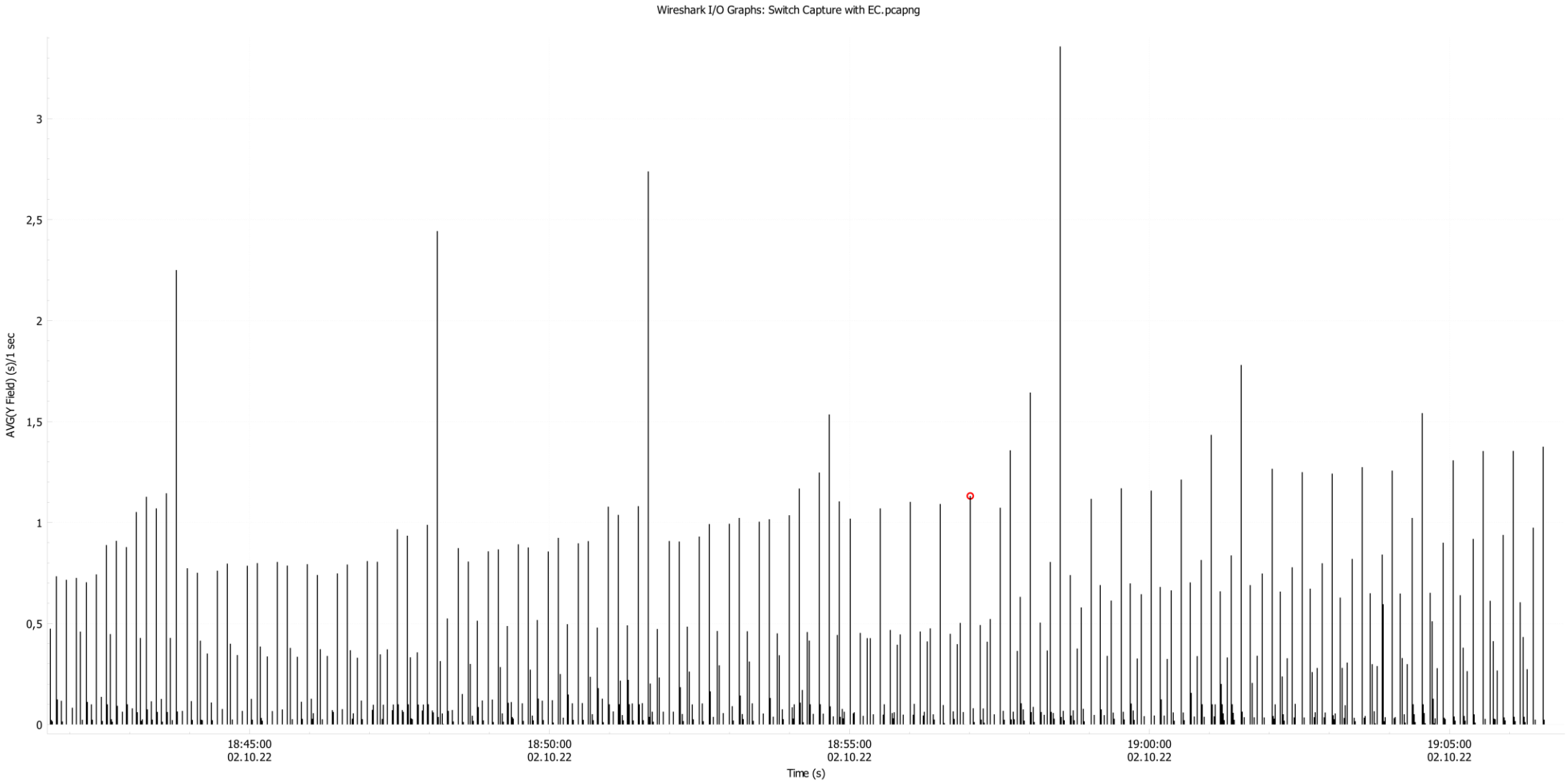

Figure 14: Average latencies in the TCP stream of the network traffic with the EC algorithm.

6. Discussion

Substantiated by the results of QoS metrics from Wireshark, the impact of EC was observed to be a reduction of ‘340.9 bits/s’ for bandwidth usage and ‘0.437 seconds’ for latency. It is noted that, the impact is a reasonably small amount which is due to several factors. The exchanged data in the testbed is objectively a minor amount compared to real-life applications, because it consists of only twelve data tags transmitted via GOOSE and enveloped in IEC 61850 MMS and XMPP headers according to IEC 61850-8-2. This can be recognized from the bandwidth usage without execution of the EC algorithm which is slightly over ‘96’ Kbits/s.

As previously explained in section 5.2, the latency is calculated based-on the time delay between consecutive packets in a single TCP stream, between each client and the XMPP server. Wireshark provides this important measurement because it represents the RTT for a data packet in a TCP conversation between client/server. This value is not an E2E latency “between PC-1 and PC-3”, it is rather an E2M latency between ‘PC-2’ and either ‘PC-1’ or ‘PC-3’. However, it provides a measure for the latency in the network.

7. Conclusions

The chapter presented an analysis of IEC 61850-8-2 XMPP communications in the developed testbed using Wireshark packet-capturing software. Initially, the data exchange and cybersecurity mechanisms of the XMPP standard were examined. Thereafter, a description was provided for the data fusion algorithm, which was developed to reduce the amount of transmitted data over the network. The simulation captions of the EC gateway model were illustrated to verify the impact of the algorithm to reduce the transmitted data tags from twelve to six. Furthermore, a Wireshark-based evaluation of the EC algorithm’s impact on the communication QoS was outlined. The methodology of the evaluation consisted of capturing data traffic for an equal period of time with and without execution of the EC algorithm.

Wireshark I/O Graphs and Microsoft Excel were applied to analyze bandwidth usage and latency for both test cases and the results were provided. The impact of executing the EC algorithm was observed. From the analysis, it was noticed that, the impact was comparatively small due to the limited capacity of the lab-scale testbed. However, the EC impact can be scaled to real-life applications where large data are being exchanged and transmitted over the network.