Synthesizing TCP Data Traffic from Industrial Networks for Simulations

Tomas HEGR, Leos BOHAC

Department of Telecommunication Engineering, Faculty of Electrical Engineering,
Czech Technical University in Prague, Technicka 2, 166 27 Prague, Czech Republic

tomas.hegr@fel.cvut.cz, bohac@fel.cvut.cz

DOI: 10.15598/aeee.v13i5.1501

Abstract. *In this paper, authors deal with a problem of an impaired TCP stream reconstruction from a real-world captured data. The goal is to obtain an original application data. The data are synthesized to be used as an input for a traffic generator. Authors describe a way to solve specific problems at transport and application layers during the reconstruction of an impaired TCP stream. The traffic reconstruction is oriented to IEC 60870-5-104 protocol on top of TCP. The evaluation of proposed algorithms shows that it is possible to estimate original time dependencies between received and dispatched messages with high accuracy.*

Keywords

IEC 60870-5-104, simulation, smart grids, TCP, traffic generator.

1. Introduction

Simulations have been one of the most favored ways of verifying new architectures and protocols in data networks for many years. Although many simulation tools and environments are used on daily basis, some fundamental problems preserve. When a simulation model is designed, it is important to consider what is the eligible input data for the simulated scenario. In the context of communication systems, the simulation input is generally produced by a traffic generation model. Although the generation model is often seen as a single entity, it is not atomic and can be decomposed to the simulation component providing the traffic generation and a description of the generated traffic.

Traditionally, the traffic generation model is stochastic, but there exist simulations where a deterministic model is preferable. When the deterministic simulation approach is applied onto a limited problem area, it can deliver precise results valuable for a network troubleshooting or forensic analysis. On the other hand, in case the number of simulated traffic flows is excessive, the deterministic traffic generator can become an obstacle due to its scalability.

Illustrative applications suitable for deterministic traffic models are implemented in some areas of the Smart Grid concept. Smart-Grid applications out of the primary mission-critical control prefer reliability to latency and use TCP (Transport Control Protocol), which provides a connection-oriented, reliable, in-sequence, byte-stream service [6]. A common example is a SCADA (Supervisory Control And Data Acquisition) regularly probing a Remote Terminal Unit (RTU) for operational data. The SCADA establishes a session irregularly, and duration of each session is varying in time. As each ISO/OSI lower layer, including TCP, is usually simulated according to the defined model, the only traffic description needed is at the application layer. However, this is very often the most problematic part, to match real traffic patterns. While requests generated by traditional user applications is common to simulate, closed industrial applications are specific and can be limited only to a single anonymous traffic capture, because of its confidential nature.

The simulation model of lower ISO/OSI layers including TCP is already implemented in most simulators, but the traffic generation at the application layer is limited. It turns out that the creation of a traffic model based on a real-world captured data is challenging, especially, if it comes to uncommon protocols. Moreover, the captured traffic in the area of industrial networks is often heavily burdened by transmission errors, and it has to be purged of retransmissions, out-of-order packets, etc.

The main goal of our traffic analysis is to describe the deterministic traffic model of a particular TCP stream, and thus, synthesize the traffic for any simulation scenario. This model can be imagined as a traffic description set containing instructions about segment lengths, precise dispatch times and logical dependencies. Considering the place of a capturing point in the network, the estimation of parameters mentioned above is a hard task. In this paper, we address several problems related to the process of synthesizing traffic description from data captured in a real-world industrial network. We focus particularly on IEC 60870-5-104 protocol [4], which is still one of the most deployed protocols in power-engineering for a remote control operations.

The paper is structured as follows:

- In Section 2. , we present related research works and standards.
- Section 3. details challenging problems.
- Section 4. closes up captured stream specifics.
- Section 5. describes our approach to selected problems.
- In Section 6. , achieved results are presented and we conclude our paper by summary in Section 7.

2. Related Work

Since the problem of the traffic analysis is complex and pervades several layers, it requires knowledge of different protocols and techniques. Beside the application layer protocol IEC 60870-5-104 standardized in [4], the fundamental is the TCP specification published in RFC 793 [13]. Most of the work on captured data relies on proper data preprocessing. In the context of the deterministic generator, it means to approximate the time when the packet is to be dispatched and to reorder or drop retransmitted and out-of-order packets.

Determining when the packet was originally dispatched, if the captured data are collected by an unknown middle-box, is a task often tackled by authors in the related area of the passive TCP Round-Trip Time (RTT) measurement. In recent publications, researchers frequently grounds their estimations in the Timestamp extension introduced in RFC 1323 [14]. For example, authors of the following publications implemented methods based on the Timestamp extension [12], [9]. Even though timestamps are very useful when determining the dispatch time, the required extension is often not incorporated in the TCP implementation at simple RTUs. Moreover, the published techniques traditionally neglect part of the latency and consider the RTT as time it takes to the client's outgoing TCP packet to be answered by the server right on the middle-box. This is only true if we can assume that one part of the network split by the middle-box evinces a significantly lower latency than the other part. This approach was attempted by authors in [8].

Focusing on the data preprocessing problem, i.e. packet retransmission, out-of-order packets and other transmission disturbances, it is necessary to understand packet's meaning in the context of the TCP flow. Authors suggest to track the connection state using a finite transition-state model as it is proposed in [8], [7], but such solution is complex. Another approach published in [1] provides a more straightforward solution. It is based on a basic packet ordering according to sequence numbers and consequent identification of a *hole* in the communication. Although the proposed algorithm is simple it gives fast and accurate results in most cases.

The simple algorithm was incorporated in a traffic generator called Swing [11]. This traffic generator observes captured traffic and tries to play it back in a way that the resulting packet trace realistically match the characteristics of the original trace. The similar approach was implemented in RENETO traffic generator [2] which is aimed to the simulation environment OMNeT++ [10]. Although the latter generator is not the only traffic generator for the OMNeT++ environment, which is the subject of our interest, it is as the only one dedicated to the reproduction of the real captured traffic. However, the generator is focused on the statistical description of the captured traffic which does not correspond to our goal.

3. Problem Definition

Synthesizing the traffic description from data captured in an unknown network is a complex task often without any way of verification. To fulfill requirements of the deterministic traffic model, it was necessary to reconstruct a time and logical dependencies of messages at the application layer. The TCP stream reconstruction was limited by the following input conditions:

- The data was captured at an arbitrary point between client and severs.
- The captured traffic contains IEC 60870-5-104, i.e. it is built on top of TCP/IP.
- The network topology is unknown.
- The captured traffic may be impaired.

As the capturing interface could be placed anywhere between TCP client and server, it was not possible to utilize any knowledge of the network topology during the TCP stream reconstruction. The method had to be based only on characteristics of the TCP and IEC 60870-5-104 protocol. We have identified three main tasks to accomplish the traffic reconstruction:

- Clean the capture of retransmitted packets and rearrange out-of-order packets.
- Identify logical dependencies between application messages and if necessary to reorder these messages.
- Estimate time when the particular application message was sent and received in context of its logical dependency.

The traffic generator model requires four parameters to generate a particular, namely sender, message length, originating message causing generation of the message (logical dependency) and relative time difference to its originator after which the message is to be dispatched (time dependency). However, only the length of messages and sender are known directly. Two remaining parameters have to be estimated as a result of the analysis in Section 4.

3.1. Split Communication Domain

Even though traffic can be captured in distributed manner on interfaces of communicating or interconnecting devices, most frequently the capture is recorded only on a single interface of an interconnecting network device. This setup is common for network monitoring stations, often integrated into regular routers, bridges, etc. Capturing on the interconnecting device, from now on referred to as Monitor, always leads to a split communication domain. As is showed in Fig. 1, the split results in two network segments, which are conventionally called upstream and downstream depending on the placement of the TCP client and server in the network. In our case, the SCADA server resides in the upstream segment and RTU is in the downstream path.

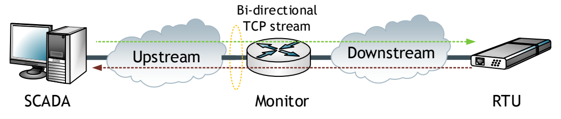


Fig. 1: The communication domain is split by the Monitor to two segments with dissimilar transmission conditions affecting the RTT estimation.

Having information only from the Monitor, it is not possible exactly determine the end-to-end delay between both communicating devices. Moreover, due to the nature of upstream and downstream paths, where the bandwidth can be dissimilar, the traditional passive RTT estimation is not enough to satisfy analysis requirements. The RTT can vary significantly during the time of the connection depending on local transmission conditions. As was mentioned before, TCP extension is also not possible to use since many RTU still implement simple old TCP algorithms. As the proper knowledge of a delay between RTU and SCADA is necessary for further investigation of TCP retransmissions and dispatch times, we have decided to implement dynamic RTT-to-ACK estimation throughout the whole capture. The RTT-to-ACK is a time interval between the TCP segment and its corresponding acknowledgment are captured at the Monitor. The algorithm is described in the analysis part in Subsection 5.2.

3.2. TCP Retransmissions

One of the main challenges in the TCP stream reconstruction problem is to deal with TCP retransmissions. When the TCP retransmission occurs, it disrupts the ordering or timing of data at the application layer. At first, to deal with retransmissions, it is necessary to identify them together with side effects accompanying retransmission from the Monitor perspective, e.g. duplicated acknowledgments (DUP-ACK) and missing segments. Primarily, following tasks have to be solved at transport layer in both directions before it the analysis of captured data for application dependencies:

- Selection of retransmitted packets to drop and to keep for the further analysis.
- Reordering of retransmitted out-of-order packets.
- Selection of pairs of retransmitted packet and its duplicated acknowledgment to drop.

4. Stream-Specific Constrains

We have investigated several available traffic captures containing dozens of TCP streams of IEC 60870-5-104. Since the TCP implementation at RTUs is often only basic TCP Reno, the analysis is in some parts implementation specific. Main observations resulting from the available captures are following:

- Not all streams are properly started by the SYN, SYN-ACK, ACK sequence and properly ended. Some streams streams are time-outed.
- IEC 60870-5-104 messages are not segmented by TCP to more packets.
- Most of the captured messages is directly followed by empty acknowledgments.

- Delayed acknowledgments are rare.
- All transmitted messages has set up the TCP push flag.
- Some streams contain retransmissions originated in the application layer.

After retransmissions were identified using expressions detailed in Subsection 5.1. , it turned out that the data for the application layer are affected in following cases:

- From the Monitor perspective, a TCP segment was acknowledged before its retransmission occurs. This means that the TCP segment was successfully received and an ACK was sent by the receiver, but the ACK was lost after it passed the Monitor. The sender retransmit the packet after reaching the Retransmission Timeout (RTO) waiting for the ACK again. In such case, we simply take into account only the first occurrence of the TCP segment and drop all retransmissions including DUP-ACKs.
- There is one or more retransmissions detected with an ACK following the retransmission sequence. The retransmitted TCP segment was lost after it passed the Monitor at least once. In this case, it is necessary to decide which packets of the sequence are to be dropped or accepted. The selection process is detailed in Subsection 5.3.
- When TCP segment with a sequence number lower than already passed segment from the same sender, it is considered to be out-of-order. In such case, the first packet was lost before it passed the Monitor. This out-of-order TCP segment has to be shifted to different place in time. The algorithm of the selection and time shift process is described in Subsection 5.4.

Cases above covered all cases of retransmissions occurring in our captured data which does not mean that these cover all possible TCP states. Our goal was to deal with the identified issues and not to produce a universal tool for the TCP traffic reconstruction. Due to the already mentioned complexity of two final state machines at two protocol layers we postponed advanced time shifts to out future work. The main algorithmic parts of the traffic reconstruction process are described in following subsections.

5. Algorithm Designs

The traffic reconstruction process is comprised of several steps. At first, the captured data is reassembled to a matrix containing parameters for the following analysis. Subsequently, the retransmitted packets are identified (Subsection 5.1.), RTT-to-ACK is estimated for both network segments (Subsection 5.2.), retransmitted packets to drop are selected (Subsection 5.3.) or reordered (Subsection 5.4.), and eventually the dependencies between IEC 60870-5-104 messages are defined (Subsection 5.5.).

5.1. Identification of Retransmitted Packets

We have inspired approach in the work [1] and based the retransmission identification on the detection of holes in sequence numbers. Since the sequence numbers from both communication sides have to be monotonically increasing with the number of sent bytes, the identification of TCP segments standing outside of the sequence is straightforward. The basic idea is depicted in Fig. 2.

Since necessary data was structured into a matrix, we were able to identify retransmissions using basic column operations. At first we filtered packets from a time ordered set of packets Eq. (1) for one direction depending on a source port as in Eq. (2|) for the client side and in Eq. (3) for the server side. Subsequently, we filtered out three sets of packets for each of the communicating sides. This filter is based on difference of TCP sequence numbers and lengths of TCP segment in shifted column as is expressed in Eq. (4), Eq. (5) and Eq. (6). Using this approach, it is obtained a set of retransmitted packets C_{ret}, a set packets C_{pnc} where the previous segment was not captured and finally a set of DUP-ACKs C_{dack}. The same process was applied on the server side packets.

$$P = \{p_0, p_1 ... p_n\}, \quad p_i^{ctime} \leq p_{i+1}^{ctime}; i \in \langle 0; n-1 \rangle, \tag{1}$$

$$C = \{p \in P \mid p^{src} = client\}, \tag{2}$$

$$S = \{p \in P \mid p^{src} = server\}, \tag{3}$$

$$,C_{ret} = C \mid_{(C_i^{snum} - (C_{i+1}^{snum} + C_{i+1}^{slen})) < 0}, \quad i \in \langle 0; |C| \rangle, \tag{4}$$

$$C_{pnc} = C \mid_{(C_i^{snum} - (C_{i+1}^{snum} + C_{i+1}^{slen})) > 0}, \quad i \in \langle 0; |C| \rangle, \tag{5}$$

$$C_{dack} = C \mid_{(C_i^{anum} - C_{i+1}^{anum} = 0) \wedge (C_i^{slen} = 0) \wedge (C_i^{fin} \neq 1)}, \quad i \in \langle 0; |C| \rangle. \tag{6}$$

Each upper index express an attribute of the particular object as follows: p^{ctime} stands for packet's captured time, p^{src} is packet's source IP address, C^{snum} is TCP sequence number, C^{slen} is TCP segment length, C^{anum} is TCP acknowledgment number and C^{fin} stands for TCP FIN flag.

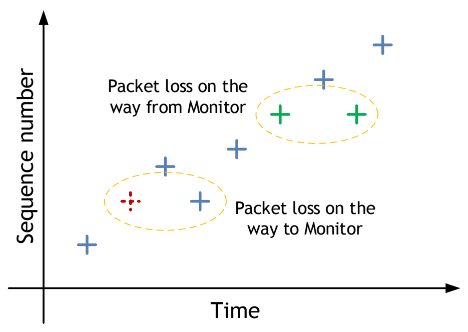

Sequence number
Packet loss on the way from Monitor
Packet loss on the way to Monitor
Time

Fig. 2: The undisturbed TCP sequence numbering should be monotonically increasing. Detecting miss-ordered sequence numbers identifies TCP retransmissions.

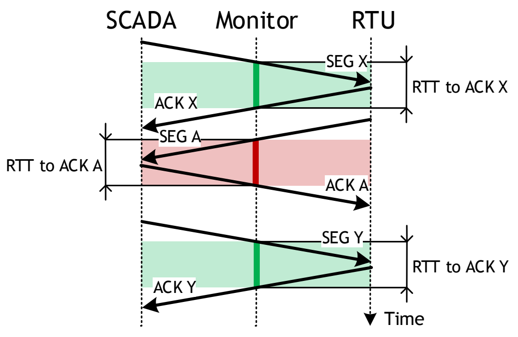


Fig. 3: The RTT-to-ACK sequence of time windows, where green windows are for the downstream and red windows are for the upstream.

5.2. RTT-to-ACK Estimation

Having identified the retransmitted packet, it is possible to compute the RTT-to-ACK delay throughout the whole capture dynamically. Iterating through the captured packets, we simply compute a time difference between a TCP segment and its ACK coming back as a response to the sender when passing the Monitor. The algorithm skips identified retransmissions. As there was a great amount of empty ACKs, i.e. segments without any payload, coming back from the receiver in both directions immediately after the TCP segments, we could limit the algorithm only for such RTT-to-ACK delays with empty ACKs. This approach removes an error caused by the time, which is needed by the receiver to process the application message.

The final product of the RTT-to-ACK is a sequence of time windows for both upstream and downstream network segments. It can be interpreted as in the Fig. 3. We did not involve packet lengths in this algorithm, as most of the packets were about the same length.

5.3. Selection from Retransmitted Packets

When there are one or more identical retransmitted TCP segments captured one by one and followed only by an ACK, it is necessary to decide to which of them the ACK was originally assigned. To deal with this problem, we took into consideration the RTT-to-ACK time windows reflecting conditions on the transmission channel in the particular time domain. Since the analysis is made off-line, it is possible to incorporate not only the time windows before the retransmission occurs but also those from the future.

The Alg. 1 is based on the parameter-scaled RTT-to-ACK time window closest to the first occurrence of the retransmission. The parameter σ determine a number of time windows accepted for a mean RTT-to-ACK. This value is substituted from the time when the ACK was captured and then the closest of the retransmitted packets is selected for further analysis. Others are dropped. The number of considered RTT-to-ACKs is limited by τ in time and by parameter ϕ in minimum number of time windows.

Algorithm 1 Retransmission selection.

Require: Set of retransmitted packets R, RTT-to-ACK times RA_{time} for sender, σ scaling factor, τ max one side time limit, ϕ min packet limit.
Ensure: Index of the selected packet i.

```
1:  $RA_{closest} = min(|RA_{time} - R.first_{time}|)$
2:  $w_{init} = RA_{closest} \cdot \sigma$
3:  $RA = RA \in ((R.first_{time} - w_{init}), (R.last_{time} + w_{init}))$
4:  $w_{RAs} = RA.last_{time} - RA.first_{time}$
5:  $w_{\tau} = (R.last_{time} + \tau) - (R.first_{time} - \tau)$
6:  if $w_{RAs} > w_{\tau}$ then
7:      $RA = RA \in w_{\tau}$
8:  end if
9:  if $|RA| < \phi$ then
10:     $RA = \phi closestRA$
11: end if
12: $t = ACK^{R}_{time} - mean(RA)$
13: return $i = minindex(|t - R_{time}|)$
```

5.4. Packet Reordering

In case the TCP segment was lost before it passed the Monitor, the retransmitted one can be captured for the first time after a segment with higher sequence number already passed the Monitor. The retransmitted TCP segment is to be placed in before the first occurrence of the following segment, which captured time is t^{source}_{prev}. This is the only case where we attempted to do a time shift during the packet reordering. As it is easy to find out a minimal time delta δ_{min}^{source} between two consecutive packets from a common sender, we estimated the time for the out-of-the-order TCP segment as $t_{prev}^{source} - \delta_{min}^{source}$.

5.5. Message Dependencies

Finally, last two algorithms describe the way a particular IEC 60870-5-104 message depends on its originating message in both time and logical domain. At first, it is necessary to estimate the time when a message was sent and received on both communicating sides. This can be done again using a column shift method on a newly created column in the original matrix. As we assume that both TCP the bi-directional flow is passing the same capturing interface and the path in a network is in both directions symmetrical, we for the simplicity approximate the end-to-Monitor delay t_{etm} as a half of RTT-to-ACK which is closest to the investigated TCP segment. The dispatch time of the message is then estimated as $t_d = t_{cap} - t_{etm}$, where t_{cap} is a time when the TCP segment was captured (including the reordering). Similarly, the time of the message reception is estimated as $t_r = t_{cap} + t_{etm}$.

The second step is to determine logical dependencies between messages. Since the IEC 60870-5-104 uses similar mechanism to TCP with sent/to-receive counters, the relations are simply identified for numbered messages (type I). However, the IEC 60870-5-104 does contain also unnumbered messages (type U) and supervisor messages (S). For this reason, we implemented a simple decision algorithm in Alg. 2.

Algorithm 2 Message dependencies.

```
Require: Investigated message $m$. Set of captured
    messages $M$.
Ensure: Originating message $o$.
 1: $\alpha = (M^{src} \neq m^{src}) \wedge (M^{t_{cap}} < m^{t_{cap}})$
 2: if $m^{type} == U$ then
 3:     $o = last(M|_{\alpha})$
 4: end if
 5: if $m^{type} == S$ then
 6:     $o = last(M|_{\alpha \wedge (M^{tx} = m^{rx} - 1)})$
 7: end if
 8: if $m^{type} == I$ then
 9:     $o = (M|_{\alpha \wedge (M^{rx} = m^{tx}) \wedge max(M^{rx})})$
10: end if
11: return $o$
```

Although the Alg. 2 always led to proper results with logical dependencies, it was necessary to correct some results in the case of the time dependencies. The time difference between the originator t_r and its successors t_s could give the overlapping timestamps. Since the message cannot be sent before its logical originator was received, we have implemented time barriers. In case of the overlapping times, the time difference of the successor dispatch time is changed to zero. Time barriers are also implemented to check estimated times against captured times. As the message, depending on the communicating side, cannot be received or sent before it was captured. In such case, the dispatch time is shifted in the middle of related captured times.

6. Evaluation

Each algorithm described in previous Section 5. was integrated into a complex application to analyze TCP streams from a real-world capture containing the IEC 60870-5-104 communication. Since it was not possible to verify the proposed approach on a capture from the undefined network topology without exact information from both communicating sides, we decided to base the evaluation on simulations. The evaluation focuses mainly on the domain of time dependencies.

The simulation model was designed as an industrial network with a point-to-point low-bandwidth channel for telemetric operations. Even though the channel is usually burdened with high Bit Error Rate (BER) in such setups, it is often shared by more TCP streams in parallel. The network model depicted in Fig. 4 consists of two connection types. First one, which is placed between Monitor and Routers, is a low-bandwidth channel with bandwidth 14 kbps and BER 10^{-4}. The second one, placed between Routers and end devices, is the standard FastEthernet with bandwidth 100 Mbps and BER 10^{-10}. The payload of inspected TCP streams was designed according to patterns found in the real-world captured IEC 60870-5-104 traffic. In our scenarios, the connection was always initiated from RTU sending bulk application data [3] in predefined times towards the SCADA server, which responded at least by 6 bytes of application data with probability varying from 0.3 to 0.5. To make the simulation more realistic, we loaded shared links by 5 concurrent TCP streams transferring random block data. Each simulation scenario was repeated one hundred times.

The analysis is based on three captures produced on SCADA, RTU and Monitor at each simulation run. All captures were recorded at devices with absolutely synchronized system times, and thus, capture time of each captured frame had the same initial time point. Following estimations are based clearly on the capture from the Monitor, as in the real-world case. Remaining captures were used only for comparison purposes. Results characterizing TCP streams are shown in Tab. 1. As one can see, the TCP stream length is varying significantly for different streams. This is due the several factors. At first, the capture itself had limited length, and at second, some streams were during

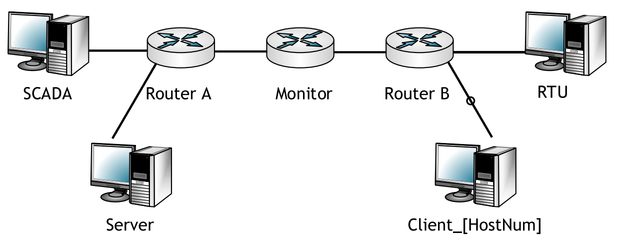


Fig. 4: Simulation topology of industrial network with low-bandwidth connections between Routers and Monitor.

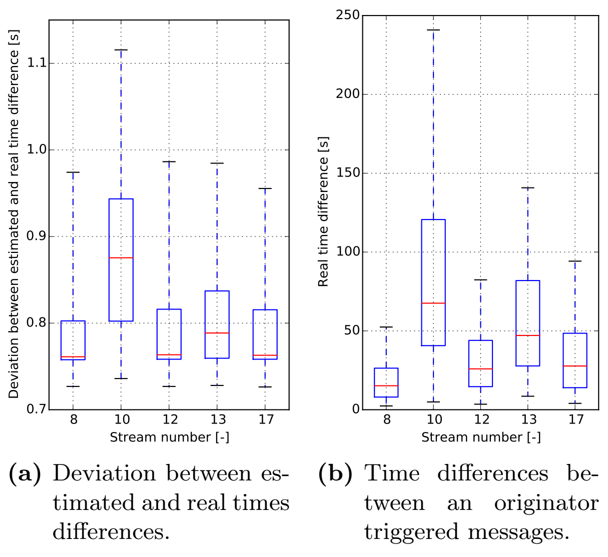
the capturing disconnected and repeatedly established. It is also notable that a total number of packets is in some case almost two times higher than the total number of transferred application messages (RTU and SCD messages).

Tab. 1: Rounded mean values characterizing TCP streams.

Id	Length [s]	Total pckts. [-]	Re-order [-]	Drop [-]	RTU msg. [-]	SCD msg [-]
8	6352	2680	10	69	1073	420
10	2634	406	3	33	87	49
12	1498	419	4	18	163	86
13	1832	298	4	15	102	48
17	4352	1080	12	52	386	126

At first, we focused on a time difference between two messages representing a SCADA control command and RTU responses, and vice versa. As the simulated data had no logical dependency defined at the application layer, we have determined the originator, i.e. the message causing dispatch of application triggered messages, as the closest TCP segment with non-zero length and acknowledgment number lower than message to be dispatched. Due to this step, all messages had originator preceding their dispatch time which correlates with the observed behavior of the IEC 60870-5-104 protocol. The results of deviations between estimated and real time differences obtained from simulations are depicted in Fig. 5a. If we compare the mean deviation of the particular TCP stream with the real-time difference between messages showed in Fig. 5b, we can find that the relative deviation is up to 4 %. This is an acceptable level for most simulation purposes.

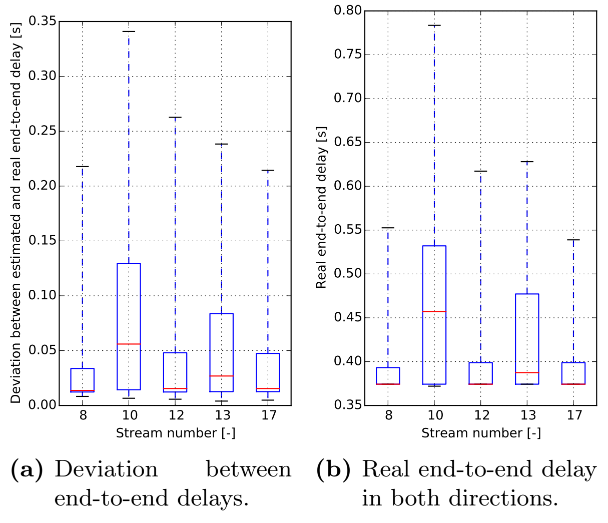
At Second, we attempted to evaluate an end-to-end delay as a second product of the time estimations based on the dynamic RTT-to-ACK evaluation. Although this is not directly related to the produced traffic description for the traffic generator, it illustrates the accuracy of the proposed approach. Results showing deviations between the estimated and real end-to-end delays are depicted in Fig. 6a. If we compare those results with a corresponding mean end-to-end delays showed in Fig. 6, we can see that the relative deviation reaches 12 % in worst case. Even though this is a higher number than in the case of time dependencies, it is still tolerable in the scope of this article.



(a) Deviation between estimated and real times differences.

(b) Time differences between an originator triggered messages.

Fig. 5: Comparison of real time differences between originator and dispatched messages obtained from end devices depicted in Fig. 5b with their deviation showed in Fig. 5a. A red line stands for mean value, a blue box shows first and third quartile and whiskers are placed on 5 % and 95 % borders.



(a) Deviation between end-to-end delays.

(b) Real end-to-end delay in both directions.

Fig. 6: Deviation of real end-to-end delays and estimated end-to-end delays in Fig 6a shows the accuracy of the proposed approach in comparison to real end-to-end delays analyzed in both directions and depicted in Fig. 6b.

7. Conclusion

The traffic generators used in simulation environments often suffer by an insufficient real-world traffic sources. Due to this reason, we decided to create an analyzer of the IEC 60870-5-104 protocol with the goal to synthesize a captured traffic to form a description file, which can be used as an input for traffic generators. Proposed algorithms show a possible way how to deal with the reconstruction of the original application data from an impaired TCP stream.

The presented approach, which is based on the dynamic evaluation of the RTT-to-ACK delay, proved its usability when estimating time dependencies for generated applications messages. Simulation results show that the relative deviation between estimated and real end-to-end delays reaches up to 12 %. Even more encouraging results were obtained in case of relative deviation between estimated and real time differences reaching in average only 4 % at investigated TCP streams.

Since we understand that the analysis is now limited to the specific TCP implementation, we plan to extend it for other implementations and to verify our results in the field as well.

Acknowledgment

This work was supported by grant no. VG20132015104, and by grant no. SGS13/200/OHK3/3T/13.

References

[1] BENKO, P. and A. VERES. A passive method for estimating end-to-end TCP packet loss. In: *Global Telecommunications Conference 2002 (GLOBECOM '02)*. Taipei: IEEE, 2002, pp. 2609–2613. ISBN 0-7803-7632-3. DOI: 10.1109/GLOCOM.2002.1189102.

[2] GEYER, F., S. SCHNEELE and G. CARLE. RENETO, a realistic network traffic generator for OMNeT++/INET. In: *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques*. Brussels: ICST, 2013, pp. 73–81. ISBN 978-1-4503-2464-9.

[3] HEGR, T. *Simulation patterns of the RTU traffic*. 2015.

[4] IEC 60870-5-104. *Transmission protocols-Network access for IEC 60870-5-101 using standard transport profiles*. Geneva: IEC, 2006.

[5] JIANG, H. and C. DOVROLIS. Passive estimation of TCP round-trip times. *ACM SIGCOMM Computer Communication Review*. 2002, vol. 32, iss. 3, pp. 75–88. ISSN 0146-4833.

[6] LEON-GARCIA, A. and I. WIDJAJA. *Communication networks: fundamental concepts and key architectures*. Boston: McGraw-Hill, 2000. ISBN 00-702-2839-6.

[7] LU, G. and X. LI. On the correspondency between TCP acknowledgment packet and data packet. In: *Proceedings of the Conference on Internet measurement - IMC '03*. New York: ACM Press, 2003, pp. 259–272. ISBN 1-58113-735-4. DOI: 10.1145/948205.948239.

[8] SCHIAVONE, M., P. ROMIRER-MAIERHOFER, F. RICCIATO and A. BAIOCCHI. Towards Bottleneck Identification in Cellular Networks via Passive TCP Monitoring. *Ad-hoc, Mobile, and Wireless Networks*. 2014, vol. 8487, no. 1, pp. 72–85. ISBN 978-3-319-07424-5. DOI: 10.1007/978-3-319-07425-2_6.

[9] STROWES, S. D. Passively Measuring TCP Round-trip Times. *Queue - High-frequency Trading*. 2013, vol. 11, iss. 8, pp. 50–61. ISSN 1542-7730. DOI: 10.1145/2523426.2539132.

[10] VARGA, A. The OMNeT++ Discrete Event Simulation System. In: *Proceedings of the European Simulation Multiconference (ESM'2001)*. Prague: ESM, 2001, pp. 1–65. ISBN 1-56555-225-3.

[11] VISHWANATH, K. V. and A. VAHDAT. Realistic and responsive network traffic generation. In: *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '06)*. New York: ACM Press, 2006, pp. 111–122. ISBN 1-59593-308-5. DOI: 10.1145/1159913.1159928.

[12] HAIJIN, Y., K. LI, S. WATTERSON and D. LOWENTHAL. Improving passive estimation of TCP round-trip times using TCP timestamps. In: *Proceedings of IEEE International Workshop on IP Operations and Management*. Beijing: IEEE, 2004, pp. 181–185. ISBN 0-7803-8836-4. DOI: 10.1109/IPOM.2004.1547614.

[13] RFC 793: Transmission Control Protocol. *Internet Engineering Task Force (IETF)* [online]. 1981. Available at: `http://www.ietf.org/rfc/rfc793.txt`.

[14] RFC 1323: TCP Extensions for High Performance. *Internet Engineering Task Force (IETF)* [online]. 1992. Available at: `http://www.ietf.org/rfc/rfc1323.txt`.

About Authors

Tomas HEGR received his M.Sc. in computer science at the Czech Technical University in Prague in 2012. He participates in teaching activities at the department of Telecommunication engineering. His research interests involve industrial networks based on Ethernet and Software-Defined Networking in all research areas.

Leos BOHAC received the M.Sc. and Ph.D. degrees in electrical engineering from the Czech Technical University, Prague, in 1992 and 2001, respectively. Since 1992, he has been teaching optical communication systems and data networks with the Czech Technical University, Prague. His research interest is on the application of high-speed optical transmission systems in a data network.