Automated Testing of Networked Systems Reliability

Michal Rozsíval
irozsival@fit.vut.cz
Faculty of Information Technology, Brno University of Technology, Czechia

Abstract

The reliability of a network is a crucial requirement for systems such as IoT, client-server, or cloud-based solutions. Unfortunately, real networks cannot be assumed to be fault-free, especially when considering various hardware problems, performance issues, or malicious attacks. Testing networked systems should therefore include evaluating fault tolerance under various network conditions. The paper presents a doctoral research project on automated verification of networked systems using fault-attack injection using a derived model of network communication.

CCS Concepts

• **Computer systems organization** → **Reliability**; • **Software and its engineering** → **Software testing and debugging**.

Keywords

Networked systems, testing, fault/attack injection, network model.

ACM Reference Format:
Michal Rozsíval. 2024. Automated Testing of Networked Systems Reliability. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '24), September 16–20, 2024, Vienna, Austria.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3650212.3685559

1 Introduction

The development of networked systems commonly occurs in laboratory conditions having, e.g., negligible delays and zero losses. However, real-world network communication does not have such properties due to physical limitations, presence of other participants, etc. Therefore, non-negligible delays, losses, reordering, or (targeted) data alteration must be considered when deploying networked systems in a real-world environment [24].

Poor treatment of imperfect networking can lead to system failures that can be costly and dangerous, especially for cyber-physical or autonomous systems—e.g., remotely controlled vehicles. Therefore, developing and testing reliable networked systems must consider various network events—faults (unintentionally occurring) and attacks (purposefully created).

Problem: Testing a networked system is complex, time-consuming, and it often requires highly specialised expertise to, e.g., modify lower-level networking (hardware or software/OS-kernel-based). Moreover, the number of test scenarios rises rapidly with different network events on which the networked system must be evaluated due to their possible combinations.

Objectives: Our approach sketched in Section 2 aims to (i) streamline, accelerate, and simplify testing and verifying of networked systems; (ii) bring the development conditions of networked systems closer to real-world situations—using fault-attack-injection; and (iii) provide the means to (semi-)automatically evaluate the correctness of the communication implementation—using criteria such as expected response to various injected faults and attacks or coverage of tested edge cases.

2 Proposed Approach

The proposed approach to achieve the objectives set up in Section 1 consists of the following research tasks. First, new techniques for monitoring real-world network traffic will be proposed to facilitate the analysis of network communications (T1). Subsequently, techniques for automated deriving of network communication models using the results of the analyses mentioned above will be developed (T2). These models will be used to verify further communications (T3) with anomaly detection in network communication as a feedback loop (T4). Finally, the created models will be used to direct the creation of new, real-world-inspired network situations, which otherwise rarely arise in the development environment (T5).

T1. Monitoring and Analysis of Network Communication: An efficient and flexible technique for monitoring and analysing network communication is a crucial part of the proposed approach, given that real-world network communication is the primary source of information for further steps. Among analysed communication properties, we include (1) *general properties*—e.g., distribution of packets over time or their size; (2) *context properties*—relations between packet occurrences; and (3) *content properties*—e.g., analysis of transmitted data. Unfortunately, none of the tools listed in Section 3 monitors all these properties. Thus, this task aims to study and develop methods that cover them.

T2. Deriving Models of Network Communication: For targeted verification of networked systems, it is necessary to have a model of the typical communication patterns between nodes. Such a model will be derived by monitoring and analysing the packets exchanged between the nodes. We will review and identify suitable models and techniques for this task (cf. Section 3) and adapt them as necessary or create new approaches to meet our specific requirements. The outcome will be a set of techniques and an accompanying tool capable of automatically deriving models using the monitoring and analysis means provided by Task T1.

T3. Verification of Networked Systems: For verifying networked systems, we will concentrate on *model-based testing and runtime verification* using the communication model(s) obtained through Task T2. These techniques, along with the faults/attacks injection studied in Task T5, will be used to (1) *drive the generation of test*

ISSTA '24, September 16–20, 2024, Vienna, Austria

ACM ISBN 979-8-4007-0612-7/24/09
https://doi.org/10.1145/3650212.3685559

scenarios and to (2) *compare the behaviour of networked systems* in the presence of injected faults or attacks *against the expected behaviour defined by the model.*

T4. Anomaly Detection in Network Communication: An anomaly can be described as a situation that differs from the usual. In the context of network communication, it can manifest in terms of (1) *general properties*—e.g., a packet has a different size or type; (2) *context properties*—e.g. a missing or unexpected packet; and (3) *content properties*—a packet may contain different data, both in terms of the error caused by a faulty network component or a targeted modification caused by an attack.

The anomalies can be detected using a model describing the usual behaviour of the network communication. Such models will be obtained using techniques developed in Task T2. Anomaly detection methods described in Section 3 can identify anomalies based on statistical data in network communication. We will review and extend these methods to detect anomalies related to context and content properties of individual packets. This enhanced anomaly detection will serve as a feedback loop for automated verification of the networked system in Task T3.

T5. Influencing Network Communication: The proposed approach focuses on bringing the development conditions of networked systems closer to the real ones to facilitate their verifying using fault/attack injection. However, none of the existing tools listed in Section 3 are suitable for that purpose. Therefore, we have already started to develop our own tool NetLoiter [19, 20]. It should provide a simple way to enforce various network conditions to allow developers to have better opportunities to verify the reliability and cybersecurity of the networked systems.

3 Related Work

This section discusses state-of-the-art approaches related to the topics in the proposed approach described in Section 2.

Monitoring and Analysis of Network Communication: Multiple tools aim to monitor and analyse network traffic, including the well-known WireShark [2] and TCPDump [11]. They primarily focus on analysing individual packets, e.g., their contents or simple links within the communication protocol [8, 23]. However, they do not look for the relation between their occurrence. Even though Deep Packet Inspection tools classify packet flows to application protocols, they only support a fixed set of protocols [10]. However, to analyse a sequence of login-requests-responses-logout packets, it is necessary to examine the packet contents and create a communication context. This includes abstracting the communication and identifying relationships and sequencing between packets.

Deriving Models of Network Communication: A variety of methodologies has been proposed for modelling communication in networked systems. An approach based on FSAs is used in [3] to model system communication, but this approach is constrained to learning a fixed number of components. Process mining [29], a well-established technique for modelling event-based systems, is not considered ideal for industrial systems as it relies heavily on data quality, which is difficult to guarantee when dealing with computer networks. If the data is incomplete, inaccurate, or has confused timelines, process mining may provide too inaccurate results [7, 13]. Anomaly detection in communication modelling [15] uses a probabilistic automata-based approach. However, most communication in industrial systems is deterministic, resulting in no probabilistic transitions in the derived automaton. Lastly, the methodologies presented in [1, 5, 12] are at a research prototype stage, not sufficiently mature for deployment in real-world distributed systems.

Verification of Networked Systems: Verifying the correctness of communication is complex, especially for real-world industrial systems. Formal verification uses methods with mathematical roots to prove the system's correctness against a formal specification. However, it requires great expertise to create and interpret formal models and struggles with large-scale systems due to the state explosion [4]. Model-based testing generates test cases from behavioural models. However, the quality of the generated test cases heavily depends on the quality of the model [16], which can be challenging to obtain in practice. Runtime verification monitors system execution for compliance but must deal with collecting and analysing runtime data, performance overhead, and the possibility of missing the errors [26].

Anomaly Detection in Network Communication: Anomaly detection methods include (1) *statistical methods* needing large amounts of data; (2) *sampling methods* observing only selected values and thus having lower memory requirements; and (3) *artificial intelligence methods* using unsupervised learning or supervised learning with a communication model. See, e.g., [21, 27, 28]. However, all these detection methods primarily rely on statistical properties of a packet stream and do not consider the content of individual packets nor its semantics in the whole communication context.

Influencing Network Communication: To create new situations in network communication, an ability to interfere with the communication, e.g., discard or delay packets, is necessary. Common network monitoring and analysis tools are unsuitable for this task as they work with copies of packets. Several hardware and software tools can affect network communication [22]. Unfortunately, hardware tools are heavy-weight, usually proprietary, and with limited availability. Currently, the most common method of simulating faults on a network communication is using software implemented fault-injection. The method focuses on the effects of the faults as the faults on a link layer are hard to be controlled [17].

Many tools support injection of faults into network communication. However, they are either purpose-specific—e.g., VirtualWire [9], or general but requiring a complicated setup or experience with networking—e.g., ThorFI [6]. Based on the technology used, the tools can be divided to (1) *simulation tools*, which simulate various network situations—e.g., OMNet++ [18] modelling entire network infrastructures; (2) *tools deployed to real systems to evaluate the resulting implementation*—e.g., Netem [14, 25] allowing to add delays, packet losses, and other queuing disciplines. However, none of them meets all our expectations: they are complicated to use (Netem), do not support dynamic reconfiguration (Netem), are slow (OMNet++), do not have all the needed functionality (ThorFI), or their use is problematic and limited as they do not directly target fault-injection (VirtualWire).

Acknowledgments

Supported by the FIT BUT project FIT-S-23-8151.

References

[1] Christopher Ackermann, Mikael Lindvall, and Rance Cleaveland. 2009. Recovering Views of Inter-System Interaction Behaviors. In *Proceedings of the 2009 16th Working Conference on Reverse Engineering (WCRE '09)*. IEEE Computer Society, USA, 53–61. https://doi.org/10.1109/WCRE.2009.34

[2] Usha Banerjee, Ashutosh Vashishtha, and Saxena Mukul. 2010. Evaluation of the Capabilities of WireShark as a tool for Intrusion Detection. *International Journal of Computer Applications* 6 (09 2010). https://doi.org/10.5120/1092-1427

[3] Ivan Beschastnikh, Yuriy Brun, Michael D. Ernst, and Arvind Krishnamurthy. 2014. Inferring Models of Concurrent Systems from Logs of Their Behavior with CSight. In *Proceedings of the 36th International Conference on Software Engineering* (Hyderabad, India) *(ICSE 2014)*. Association for Computing Machinery, New York, NY, USA, 468–479. https://doi.org/10.1145/2568225.2568246

[4] Raik Brinkmann and Dave Kelf. 2018. *Formal Verification—The Industrial Perspective*. Springer International Publishing, Cham, 155–182. https://doi.org/10.1007/978-3-319-57685-5_5

[5] Paolo Milani Comparetti, Gilbert Wondracek, Christopher Kruegel, and Engin Kirda. 2009. Prospex: Protocol Specification Extraction. In *2009 30th IEEE Symposium on Security and Privacy*. 110–125. https://doi.org/10.1109/SP.2009.14

[6] Domenico Cotroneo, Luigi De Simone, and Roberto Natella. 2022. ThorFI: a Novel Approach for Network Fault Injection as a Service. *Journal of Network and Computer Applications* 201 (2022), 103334.

[7] Dusanka Dakic, Srdjan Sladojevic, Teodora Lolic, and Darko Stefanovic. 2019. Process Mining Possibilities and Challenges: A Case Study. In *2019 IEEE 17th International Symposium on Intelligent Systems and Informatics (SISY)*. 000161–000166. https://doi.org/10.1109/SISY47553.2019.9111591

[8] John Day and Hubert Zimmermann. 1984. The OSI reference model. *Proc. IEEE* 71 (01 1984), 1334 – 1340. https://doi.org/10.1109/PROC.1983.12775

[9] P. De, A. Neogi, and T.-C. Chiueh. 2003. VirtualWire: a fault injection and analysis tool for network protocols. In *Proc. of 23rd Int. Conference on Distributed Computing Systems*. 214–221. https://doi.org/10.1109/ICDCS.2003.1203468

[10] Reham Taher El-Maghraby, Nada Mostafa Abd Elazim, and Ayman M. Bahaa-Eldin. 2017. A survey on deep packet inspection. In *2017 12th International Conference on Computer Engineering and Systems (ICCES)*. 188–197. https://doi.org/10.1109/ICCES.2017.8275301

[11] Felix Fuentes and Dulal Kar. 2005. Ethereal vs. Tcpdump: A comparative study on packet sniffing tools for educational purpose. *Journal of Computing Sciences in Colleges* 20 (01 2005), 169–176.

[12] Yating Hsu, Guoqiang Shu, and David Lee. 2008. A model-based approach to security flaw detection of network protocol implementations. In *2008 IEEE International Conference on Network Protocols*. 114–123. https://doi.org/10.1109/ICNP.2008.4697030

[13] Mieke Jans and Manal Laghmouch. 2023. *Process Mining for Detailed Process Analysis*. Springer International Publishing, Cham, 237–256. https://doi.org/10.1007/978-3-031-11089-4_9

[14] Linux Manual Page. 2011. NetEm - Network Emulator. https://www.man7.org/linux/man-pages/man8/tc-netem.8.html

[15] Petr Matoušek, Vojtěch Havlena, and Lukáš Holík. 2021. Efficient modelling of ICS communication for anomaly detection using probabilistic automata. In *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 81–89.

[16] Muhammad Luqman Mohd-Shafie, Wan Mohd Nasir Wan Kadir, Horst Lichter, Muhammad Khatibsyarbini, and Mohd Adham Isa. 2022. Model-Based Test Case Generation and Prioritization: A Systematic Literature Review. *Softw. Syst. Model.* 21, 2 (apr 2022), 717–753. https://doi.org/10.1007/s10270-021-00924-8

[17] Roberto Natella, Domenico Cotroneo, and Henrique S. Madeira. 2016. Assessing Dependability with Software Fault Injection: A Survey. *ACM Comput. Surv.* 48, 3, Article 44 (feb 2016), 55 pages. https://doi.org/10.1145/2841425

[18] OMNet++. [n. d.]. Discrete Event Simulator. https://omnetpp.org/

[19] Michal Rozsíval. 2023. *Simulované vkládání chyb v síťové komunikaci*. Master's thesis. Vysoké učení technické v Brně, Fakulta informačních technologií.

[20] Michal Rozsíval and Aleš Smrčka. 2023. NetLoiter: A Tool for Automated Testing of Network Applications using Fault-injection. In *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. 207–210. https://doi.org/10.1109/DSN-W58399.2023.00057

[21] Hadiseh Safdari and Caterina De Bacco. 2022. Anomaly detection and community detection in networks. *Journal of Big Data* 9, 1 (dec 2022). https://doi.org/10.1186/s40537-022-00669-1

[22] Nadir K. Salih, D Satyanarayana, Abdullah Said Alkalbani, and R. Gopal. 2022. A Survey on Software/Hardware Fault Injection Tools and Techniques. In *2022 IEEE Symposium on Industrial Electronics & Applications (ISIEA)*. 1–7. https://doi.org/10.1109/ISIEA54517.2022.9873679

[23] Richard Sharpe, Ed Warnicke, and Ulf Lamping. [n. d.]. Wireshark User's Guide. https://www.wireshark.org/docs/wsug_html_chunked/ChAdvFollowStreamSection.html

[24] Carlos Alexandre Gouvea Da Silva and Carlos Marcelo Pedroso. 2019. MAC-Layer Packet Loss Models for Wi-Fi Networks: A Survey. *IEEE Access* 7 (2019), 180512–180531. https://doi.org/10.1109/ACCESS.2019.2958260

[25] Hemminger Stephen. 2005. Network Emulation with NetEm. In *Proceedings of the 6th Australia's National Linux Conference*. 1–8.

[26] César Sánchez, Gerardo Schneider, Wolfgang Ahrendt, Ezio Bartocci, Domenico Bianculli, Christian Colombo, Yliés Falcone, Adrian Francalanza, Srđan Krstić, JoHao M. Lourenço, Dejan Nickovic, Gordon J. Pace, Jose Rufino, Julien Signoles, Dmitriy Traytel, and Alexander Weiss. 2018. A survey of challenges for runtime verification from advanced application domains (beyond software). *Formal Methods in System Design* 54 (2018), 279 – 335. https://link.springer.com/article/10.1007/s10703-019-00337-w

[27] Marina Thottan and Chuanyi ji. 2003. Anomaly Detection in IP Networks. *Signal Processing, IEEE Transactions on* 51 (09 2003), 2191 – 2204. https://doi.org/10.1109/TSP.2003.814797

[28] Marina Thottan, Guanglei Liu, and Chuanyi ji. 2010. *Anomaly Detection Approaches for Communication Networks*. 239–261. https://doi.org/10.1007/978-1-84882-765-3_11

[29] Wil van der Aalst. 2016. *Process Modeling and Analysis*. Springer Berlin Heidelberg, Berlin, Heidelberg, 55–88. https://doi.org/10.1007/978-3-662-49851-4_3

Received 2024-07-07; accepted 2024-07-22