WIP: Emulation Framework for Byzantine Fault-Tolerant Protocols

Bastian Kupka[1], Tim Strutzenberger[1], Simon Egger[1], and Christian Becker[1]

1 Introduction

Byzantine faults capture arbitrary system failures. Nodes may crash, messages may be delayed and corrupted, or nodes may even send malicious messages to the remaining honest processes in the system. This makes Byzantine Fault Tolerance (BFT) essential for building resilient distributed systems like distributed databases [Va08] and for designing system-critical protocols like clock synchronization [DW04].

In this work, we present an in progress network emulator that aims towards a platform for testing (both generic and protocol-specific) fault-tolerance models for a wide range of network protocols. Our emulator is built upon Linux network namespaces and utilizes the NetEm queueing discipline [Ne]. This provides a generic method (with minimal required changes to the source code) to evaluate resilience against asymmetric delays, packet delay variations, packet loss, and packet corruption. We are currently working on extending the emulator to support black- and grey-box fuzzing techniques, along with a simple interface to aggregate and evaluate the test results.

2 Related Work and Problem Statement

Analyzing Byzantine fault tolerance can be a highly complex and elaborate task that involves rigorous formal analysis or formal specification languages like TLA+ [La99]. Replacing such formal methods is not one of our design goals [Di72]; Instead, we aim for an emulation platform for detecting generic and protocol-specific faults in a wide range of existing network protocol implementations, with minimal required changes to their source code.

Tools like Mininet [LHM10] or Jepsen [Je] are more closely related to this work. Mininet enables a simple emulation of networks that is often used for Software Defined Networking (SDN). Jepsen provides a framework to test safety and liveness guarantees of distributed systems (e.g., MySQL, Redis, Zookeeper) in a real deployment. In contrast, our emulation framework aims to combine the flexibility of Mininet network deployments with the ability of Jepsen to test safety and liveness guarantees.

[1] University of Stuttgart, Institute for Parallel and Distributed Systems, Germany, st177153@stud.uni-stuttgart.de; st172242@stud.uni-stuttgart.de; simon.egger@ipvs.uni-stuttgart.de; christian.becker@ipvs.uni-stuttgart.de

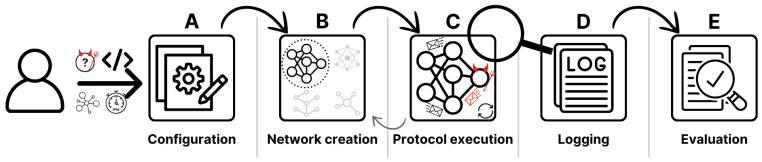

Fig. 1: Steps using our network emulator

3 Network Emulator Design

An overview of the individual steps is shown in Fig. 1. In Step A, the user configures test cases for a given network protocol implementation, i.e., in the form of an executable binary. By using Linux network namespaces to build the networks (Step B), we allow different processes to communicate via Linux sockets (e.g., UPD, TCP, or raw sockets). In Step C, each test case is run one by one. We support two complementary categories of fault models that can be configured by the test case: First, we expose the capabilities of the NetEm queueing discipline, e.g., manipulated packet delays, packet loss, or packet corruption. Second, we support a centralized controller (attacker) that has a global view of the network and has full control over all malicious processes. The controller can thereby start a generic fuzzing campaign or a protocol-specific attack. After each run, the logs of all processes are analyzed to check if any of the user-provided liveness or safety properties are violated.

Acknowledgment

This work was supported by the European Commission through the H2020 project DETERMINISTIC6G (Grant Agreement no. 101096504).

Bibliography

[Di72] Dijkstra, Edsger W: The humble programmer. Communications of the ACM, 15(10):859–866, 1972.

[DW04] Dolev, Shlomi; Welch, Jennifer L.: Self-stabilizing clock synchronization in the presence of Byzantine faults. J. ACM, 51(5):780–799, September 2004.

[Je] Jepsen. https://jepsen.io/, Accessed: 2025-01-31.

[La99] Lamport, Leslie: Specifying concurrent systems with TLA+. Calculational System Design, pp. 183–247, 1999.

[LHM10] Lantz, Bob; Heller, Brandon; McKeown, Nick: A network in a laptop: rapid prototyping for software-defined networks. In: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. pp. 1–6, 2010.

[Ne] NetEm – Network Emulator. https://man7.org/linux/man-pages/man8/tc-netem.8.html, Accessed: 2025-01-31.

[Va08] Vandiver, Benjamin Mead: Detecting and tolerating byzantine faults in database systems. PhD thesis, Massachusetts Institute of Technology, 2008.