Hatebefi: Hybrid Applications Testbed for Fault Injection

Arne Boockmeyer*, Jossekin Beilharz†, Lukas Pirl† and Prof. Dr. Andreas Polze†
Operating Systems and Middleware Chair, Hasso Plattner Institute
Potsdam, Germany
Email: *{firstname.lastname}@student.hpi.uni-potsdam.de, †{firstname.lastname}@hpi.uni-potsdam.de

***Abstract*—Hybrid testbeds are popular for testing distributed software systems, like network protocols and distributed applications, since the beginning of the 2000s. Combining physical and virtual resources for testing these networked computer systems allows to leverage the advantages and mitigate the disadvantages of either one. However, hybrid testbeds introduce novel challenges, e.g. regarding plausibility, heterogeneity, and controllability. To counter these challenges, we introduce the Hybrid Applications Testbed for Fault Injection (Hatebefi) which combines two approaches: On the one hand, Hatebefi aims to increase the plausibility by integrating hybrid testbeds with domain-specific simulators (e.g. for traffic simulation). This integration also addresses the heterogeneity of contemporary distributed applications. On the other hand, our framework allows for efficient creation and execution of complex test scenarios with a high degree of controllability by offering an event-based execution model. To demonstrate the feasibility of our approach, we implemented a basic set of arbitrarily combinable events to cover the most common scenarios. Both features combined pave the way to test distributed software systems, like Internet of Things applications involving connected vehicles or smart cities.**

***Index Terms*—hybrid, testbed, *ns-3*, domain-specific simulation, vehicular-to-everything, IoT, fault tolerance, reliability**

I. Introduction

With the advent of networked computer systems and distributed applications, developers became aware of the problems when testing applications on production networks like the Internet. Therefore, separate networks with physical nodes were set up for testing applications in controlled environments before integrating them into the production systems. While this approach protects the production systems, it is still hard to handle the manifold external influences on the test environment [1]. Furthermore, hardware testbeds cannot easily be scaled, since this requires physical resources to be acquired (e.g. network and computer hardware).

Software-based testbeds allow to mitigate the aforementioned limitations by virtualising nodes and simulating networks. As a result, less hardware needs to be acquired and less interaction with the hardware is required to allow the testing of a large-scale distributed system with many nodes. Those circumstances simplify, the composition (e.g. network topologies), parametrisation (e.g. network bandwidth and delay), scaling (e.g. number of nodes) and control (e.g. full automation) of tests and experiments. On the downside, the behaviour of virtualisations and simulations often don't match the reality in all characteristics, since they inherently incorporate abstractions and simplifications [1] [2].

The idea to combine the advantages of hardware and software testbeds to hybrid testbeds suggests itself and indeed is a popular approach. Hybrid testbeds are formed from a mixture of hardware and software resources (e.g. nodes, network connections). While the hardware resources are intended to increase the plausibility of the investigations (e.g. electromagnetic interference), the software resources are intended to maintain flexibility (e.g. scaling) and controllability (e.g. event-drivenness).

With the pervasion of Internet of Things (IoT) applications, the requirements for testbeds shift towards large, widely distributed and rapidly changing systems. We therefore propose a novel approach which unburdens the investigation of such scenarios with manageable complexity. Our vision is to create an event-based hybrid testbed connected to specialised simulators, such as for traffic. The event-based approach allows to create dynamic, flexible and well-controllable test scenarios to expose the software under consideration to fluctuating conditions. The connection to specialised simulators (like in [5]) increases the plausibility since they contribute the expert knowledge for a certain domain. For example, this offers the possibility to investigate intelligent transportation systems, where vehicles move realistically (e.g. routes, speeds, amounts) through urban areas and communicate wirelessly in an ad hoc fashion. The Hybrid Applications Testbed for Fault Injection (Hatebefi) enables test designers to encode the overall test setup (e.g. nodes and network topologies) and test scenarios (e.g. changing network conditions).

II. Hatebefi - A Hybrid Testbed

Hatebefi is implemented as a *Python* extension framework to the network simulator *ns-3*[1]. Using the framework is straightforward and does not require specialised tools: import the Hatebefi library into a ns-3 Python project and execute ns-3 with the included Python bindings. The framework offers abstractions for common tasks regarding orchestration and simulation. For features not covered by those abstractions, the native Python bindings of ns-3 can still be used.

Hatebefi consists of four major components. The first component is the node (hardware or software) orchestration which

[1] https://www.nsnam.org

abstracts node creation (e.g. operating system) and management (e.g. start, stop, freeze). The abstractions for ns-3 are implemented by the second component which provides a consistent and intuitive interface to the network simulation. The third component implements the connection to the domain-specific simulation which can be used to obtain coordinates of realistically moving nodes. The fourth component provides the events API which allows developers to create events and react to changes of the system state with fine granularity. Therewith, it facilitates control during runtime over the test cases which have been built using the three components mentioned first.

A. Node Orchestration and Network Abstraction

A requirement for Hatebefi is to combine several different back ends for Virtual Machines (VMs) and containers with hardware nodes. For both, hardware and software nodes, Hatebefi provides an abstraction layer for their creation, management and for connecting them to ns-3 effortlessly. For hardware nodes — which of course have to be reachable over the network from the host — another abstraction layer is provided, so that those can be connected to ns-3 likewise. An advantageous side effect of this architecture is that it is also possible to connect Hatebefi to remote hypervisors.

In analogy to managing nodes, there is also a layer to abstract from the details of the network configuration. Using these abstractions, creating a ns-3-managed network requires just a few lines of code.

B. Connection to a Domain-Specific Simulator

To increase the plausibility of results, Hatebefi connects the hybrid testbed to domain-specific simulators. Currently we focus on the traffic simulator "Simulation of Urban Mobility" (SUMO) [4]. Our vision is to represent every vehicle in such a simulation with a node in the simulated network. This allows us to run test scenarios like assessing the communication between wirelessly communicating cars, trains and infrastructure elements.

C. Events API

To control the test scenario dynamically during runtime, Hatebefi introduces an expressive events API. It allows the test developer to perform the operations as described in III dynamically during the simulation. It bases on three different event types which can be queued in any order:

- `after`: The `after` event waits a given amount of time.
- `when`: The `when` event takes a condition as Python lambda. Once the result of this condition matches an expected result, the processing continues. It uses an active expression implementation for Python[2] to get all dependencies of the condition.
- `check-if`: The `check-if` event takes a Python lambda expression and it only continues the processing if the lambda returns an expected value. It is also possible to make this periodic and check it in a given interval.

[2]https://github.com/active-expressions/active-expressions-static-python

The execution of the script which describes the test scenario is continued once the respective event occurs. Events can be combined arbitrarily and can hence express complex scenarios. The processing of an event can start directly when creating the event or at the beginning of the simulation.

III. Operations for Fault Injection Experiments

Currently there are several operations supported:

- Create and modify the network topology: Create new networks during the simulation, connect nodes to existing networks and disconnect nodes from networks. Also managing IP addresses is supported.
- Change network configuration: Change the bandwidth or latency of a network on the fly.
- Manage nodes: Start, stop or restart nodes and execute commands on the nodes during the simulation to adjust the configuration of the nodes.

These operations allow test designers to perform several different scenarios to test network applications especially in the perspective of fault injection. For example, it could be assessed how distributed file systems react to node failures. Or, in an IoT context, to test the reliability of data transmissions over connections with a very high latency towards the edge.

IV. Conclusion

Hatebefi is a modern hybrid testbed integrated as a framework into the network simulator ns-3. It can use different back ends for VMs, containers and hardware nodes. Currently, a proof of concept is implemented for *LXD* and hardware nodes, while the support for *Docker* is near completion. The vision to connect domain-specific simulators is not implemented yet. We expect the use either SUMO's trace files or traCI API for the connection the testbed. Hatebefi allows to test distributed applications with regard to, e.g., fault tolerance or behaviour under altered network conditions. The framework offers an events API to make authoring the test cases as flexible as possible. To increase the plausibility we connect Hatebefi to domain-specific simulators, like SUMO for traffic simulations to control the network nodes' movements realistically. Especially in the era of IoT, this opens up new ways to test distributed applications in a controlled and scaled fashion.

References

[1] A. Zimmermann, M. Gunes, M. Wenig, U. Meis and J. Ritzerfeld, "How to Study Wireless Mesh Networks: A hybrid Testbed Approach," 21st International Conference on Advanced Information Networking and Applications (AINA '07), Niagara Falls, ON, 2007, pp. 853-860.

[2] T. Miyachi, K. Chinen, and Y. Shinoda, "StarBED and SpringOS: Large-scale General Purpose Network Testbed and Supporting Software" in Proceedings of the 1st International Conference on Performance Evaluation Methodologies and Tools VALUETOOLS '06, Pisa, Italy, 2006, p. 30.

[3] H. Kim et al., "IoT-TaaS: Towards a Prospective IoT Testing Framework," in IEEE Access, vol. 6, pp. 15480-15493, 2018.

[4] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO – Simulation of Urban MObility: An Overview," in Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation, Barcelona, 2011.

[5] B. Schünemann, "V2X simulation runtime infrastructure VSimRTI: An assessment tool to design smart traffic management systems", in Comput. Netw. 55, 14, pp. 3189-3198, 2011.