Online Testing of Federated and Heterogeneous Distributed Systems

Marco Canini, Vojin Jovanović, Daniele Venzano, Dejan Novaković, and Dejan Kostić
School of Computer and Communication Sciences, EPFL, Switzerland
{marco.canini,vojin.jovanovic,daniele.venzano,dejan.novakovic,dejan.kostic}@epfl.ch

ABSTRACT

DiCE is a system for online testing of federated and heterogeneous distributed systems. We have built a prototype of DiCE and integrated it with an open-source BGP router. DiCE quickly detects three important classes of faults, resulting from configuration mistakes, policy conflicts and programming errors.

The goal of this demo is to showcase our DiCE prototype while it executes an experiment that involves exploring BGP system behavior in a topology with 27 BGP routers and Internet-like conditions (Figure 1).

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems

General Terms

Reliability

Keywords

Online testing, Fault detection, Federated and heterogeneous distributed systems

1. INTRODUCTION

The successfully deployed distributed systems typically use open interfaces, but often end up being heterogeneous due to the creation of multiple implementations. Moreover, the success drives deployment across the wide-area network, which then leads to the systems becoming federated to allow separate administrative domains to retain control over the local nodes' configuration. A prime example of such a system is the Internet's inter-domain routing, today based on BGP.

The important services these systems provide have to remain uninterrupted over long periods of time. However, the unanticipated interaction of nodes under seemingly valid configuration changes and local fault-handling can have a profound effect. For example, the Internet's routing has suffered from multiple IP prefix hijackings, as well as performance and reliability problems due to emergent behavior resulting from a local session reset.

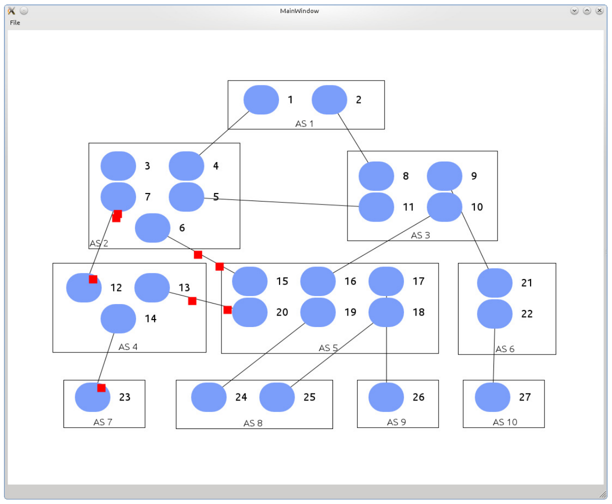

Figure 1: A graphical interface showing the execution of DiCE over a topology with 27 BGP routers.

In a position paper [3], we argued that making heterogeneous and federated distributed systems reliable is challenging because (i) the source code of every node may not be readily available for testing and (ii) competitive concerns are likely to induce individual providers to keep private much of their current state and configuration.

In an effort to increase distributed system reliability, our overarching vision is to harness the continuous increases in available computational power and bandwidth. Specifically, we argue that the system should proactively detect potential faults that can occur in it due to, for example, programming errors, policy conflicts, and operator mistakes. In this work, our goal is to explore system behavior so to find node actions that lead to faults.

The remainder of this paper is organized as follows. Section 2 provides an overview of our approach. Section 3 highlights certain prototype details and evaluation results. Finally, Section 4 discusses some related material.

2. APPROACH OVERVIEW

We propose DiCE, an approach that continuously and automatically explores the system behavior, to check whether the system deviates from its desired behavior. At a high-level, as illustrated in Figure 2, DiCE (i) creates a snapshot consisting of lightweight node checkpoints, (ii) orchestrates the exploration of relevant system behaviors across

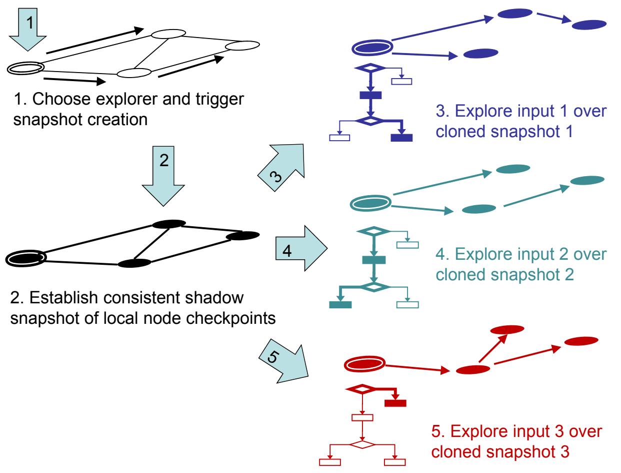
1
1. Choose explorer and trigger snapshot creation
2
2. Establish consistent shadow snapshot of local node checkpoints
3
3. Explore input 1 over cloned snapshot 1
4
4. Explore input 2 over cloned snapshot 2
5
5. Explore input 3 over cloned snapshot 3

Figure 2: DiCE systematically explores and checks system behavior over isolated snapshots.

the snapshot by subjecting system nodes to many possible inputs that exercise node actions, and (*iii*) checks for violations of properties that capture the desired system behavior. DiCE starts exploring from current system state, and operates alongside the deployed system but in isolation from it. In this way, testing can account for the current code, state and configuration of the system. DiCE reuses existing protocol messages to the extent possible for interoperability and ease of deployment.

DiCE drives exploration by using concolic execution [4] to produce inputs that systematically explore all possible paths at one node. Concolic execution[1] is an automated software testing technique that executes a program by treating the inputs to the program as symbolic. A concolic execution engine tracks the constraints of the code branches on symbolic inputs encountered during execution. For each constraint, it then queries a solver to find a value that negates the constraint and leads down to a different code path.

We face several difficult challenges in our work. Because of the federated nature and heterogeneity of the systems we target, we cannot drive system behavior from atop. DiCE explores system behavior by letting each node autonomously exercise its local actions and observe their system-wide consequences over a set of lightweight checkpoints. Second, approaches that systematically explore code paths easily run into the problem of exponential explosion of possible code paths. Three insights allow us to manage this problem: (*i*) we start exploring from current system state so that we avoid to replay a long history of inputs to explore deep in code behavior. (*ii*) We localize and focus on code that changes state (*e.g.*, message handlers). This decision allows us to quickly explore relevant code paths, whereas other code such as message parsers could be tested offline. (*iii*) We subject the node's code to small-sized inputs, and apply grammar-based fuzzing to produce a large number of valid inputs. Lastly, it is challenging to detect faults by checking for violations of properties while there cannot be unrestricted access to remote node states because the systems we target are federated. We define a narrow information-sharing interface that allows nodes to communicate the result of local state checks while preserving confidential information.

[1]A variant of symbolic execution. Concolic stands for CONCrete + symbOLIC.

3. DiCE PROTOTYPE AND EVALUATION

We have successfully integrated DiCE with the BGP implementation of the BIRD open-source router [1]. We use the Oasis concolic execution engine [4] as the basis for code path exploration. For integrating with BIRD, we first change its BGP implementation to mark certain inputs as symbolic. We choose to treat UPDATE messages as the basis to derive new inputs during exploration. For instance, the Network Layer Reachability Info (NLRI) region of the message contains the announced routes with their respective netmask lengths. We mark these as symbolic. An UPDATE message also typically carries multiple path attributes each of which is encoded as a type, length, and value fields that are also treated as symbolic. In practice, this choice allows DiCE to construct inputs that exercise BGP behavior in two dimensions: the first due to BIRD's code implementing BGP, the second as the result of the particular configuration currently in use. This is because the source code instrumentation encompasses the BIRD's configuration interpreter and so allows Oasis to record constraints for the interpreted configuration. Therefore, the explored execution paths are comprehensive of both code and configuration. Finally, we consider behaviors due to configuration changes. We treat as symbolic the condition that describes whether a route is the locally most preferred one. This allows us to systematically explore the outcome of BGP's route selection process.

Our evaluation using a set of BGP routers in a testbed with Internet-like conditions demonstrates DiCE's effectiveness, low overhead, and ease of integration with existing software written in C. Specifically, our prototype quickly detects faults that can occur due to programming errors, policy conflicts, and operator mistakes.

4. RELATED MATERIAL

Our work is on-going. We published a position paper in [3], and a paper describing our initial design and prototype is in [2]. A presentation of DiCE given at the RIPE62 meeting is available at `http://ripe62.ripe.net/archives/video/96`. A technical report in [4] fully describes Oasis, the concolic execution engine we use.

5. REFERENCES

[1] The BIRD Internet Routing Daemon. `http://bird.network.cz`.

[2] M. Canini, V. Jovanović, D. Venzano, B. Spasojević, O. Crameri, and D. Kostić. Toward Online Testing of Federated and Heterogeneous Distributed Systems. In *USENIX ATC*, June 2011.

[3] M. Canini, D. Novaković, V. Jovanović, and D. Kostić. Fault Prediction in Distributed Systems Gone Wild. In *LADIS*, July 2010.

[4] O. Crameri, R. Bachwani, T. Brecht, R. Bianchini, D. Kostić, and W. Zwaenepoel. Oasis: Concolic Execution Driven by Test Suites and Code Modifications. Technical Report LABOS-REPORT-2009-002, EPFL, 2009.