Hyperledger Umbra Internship Project 2019: Mid-term Activities Report

Summary

Hyperledger Umbra Internship 2019 has its source code current uploaded at https://github.com/raphaelvrosa/umbra
This code contains three modules (umbra-configs, umbra-orch and umbra-scenarios).

- umbra-configs: concerns the creation of *apriori* configurations to be executed by the other modules, defined by a scenario and a set of events
- umbra-orch: realizes the main functionalities of the project, when having assigned a config (output of umbra-configs) realizes the scenario deployment (via umbra-scenarios) and performs the triggering of scheduled events in the executed environment.
- umbra-scenarios: interfaces Mininet/Containernet to deploy the required scenario demanded by umbra-orch, besides of executing events, such as node up/down, link up/down, and updates in link attributes (e.g., delay, bandwidth, packet loss).

Currently, this components are coded using the python programming language. To showcase the functionality of these umbra modules, the project Hyperledger Fabric can be reproduced as an example.
As next activities, the Hyperledger projects Iroha and Indy will also be mapped to be executed as Fabric currently is. In addition, all the dynamicity of topology events, and monitoring functionalities will be included in the Umbra platform until the final-term evaluation.
As it was conducted so far, the activities of the Umbra project follow detailed in sequence.

Activities in Details

Each activity described below contains its main realizations, achievements and issues.

➔ Build reference architecture: concerns the construction of the overall architecture, the project directory structure, and the design of the following project views: logical, use case, and process.
  ◆ Achievements: design of main python modules for umbra architecture
  ◆ Issues: difficult to design each module, as no details of Hyperledger projects, and their commonalities, were specified promptly

➔ Prototyping the Containernet plugin - umbra-scenarios: the Containernet platform was utilized to build a python module to be operated via a REST interface. When

received a topology description this module instantiates it using Containernet functionalities, and replies the description of the management information of the deployed topology.

- Achievements: construction of a topology descriptor, detailing nodes/links and their attributes (e.g., vcpus, memory, bandwidth, etc). Instantiation/tear-down of topology (docker containers and links among them). Links can be specified via programmable switches (Open vSwitch - supporting OpenFlow).
- Issues: docker API is deprecated; Containernet runs in python2.7; if topology in ring, needs to specify Spanning Tree Protocol (STP); attribution of IP addresses to containers; extension of Containernet classes to support Docker execution requirements for Hyperledger projects (e.g., working dir, dns names, port bindings); Defined attribution of ARP names/addresses automatically for all nodes (i.e., no need of ARP resolutions);

➔ Prototyping umbra-orch: main module of Umbra, receives a configuration descriptor, triggers the topology deployment via umbra-scenarios, and triggers events on top of components deployed. Events can be generic to the whole topology (e.g., node up/down, monitoring, update link attribs) as well as specific to a Hyperledger project (e.g., Fabric chaincode deployment and invoke).

- Achievements: well defined component with REST interface to be called upon configuration execution request, parsing of events to be called on timely manner, plugin subsystem defined for particular Hyperledger projects.
- Issues: timely manner events are called upon start of scenario execution (from second zero on);

➔ Prototyping umbra-configs: python module that creates configurations (topology scenario and events) to be sent to be instantiated by umbra-orch.

- Achievements: Modular configuration files (i.e., topology + events = scenario); topology is extensible for particular Hyperledger project (e.g., FabricTopology builds all its configs from scratch and abstracts node functionalities)
- Issues: each Hyperledger project demands particular requirements to be expressed in events (trade-off: specific vs. generic)

➔ Construction of Examples: defines modular way of generating configuration for each Hyperledger project and its execution using umbra modules.

- Achievements: from a single configuration file, any Hyperledger project will be able to be executed in umbra. Example detains build scripts for each project dependencies (e.g., download/build/upgrade Docker images). Scripts to execute configurations starts environment from scratch and cleans scenario when stopped.
- Issues: design trade-offs where to reference/store particular configurations

Overall Checklist:

➔ Build reference architecture: OK
➔ Evaluate Fabric project with reference architecture: OK
➔ Build Stimulus: OK (not uploadded to github - still running tests)

➔ Build Monitoring: OK (not uploadded to github - still running tests)
➔ Build Dynamics: OK (not uploadded to github - still running tests)

Issues

Lack of Fabric operational documentation: details of configuration structure, events, APIs, interfaces, calls, messages structure, SDK workflows.

Plan for final-term Evaluation

Minor:

- Events triggered in particular temporal conditions, following the properties “when, during, until, repeat”.
- Run-time interface for umbra-orch: calling events on demand

Major:

- Support for Iroha and Indy
- Module umbra-mon:
- Support for more events:
- Experiment report:

Proposal for Hyperledger Global Forum 2020

Showcase Hyperledger Umbra through Fabric, Iroha, Indy demos.
Demo containing all the Umbra features: project reproductibility, analytics, events.