November 2022

Network Test Automation Based on Network Model

n/a

Network Test Automation Based on Network Model

ABSTRACT

This disclosure describes techniques for automated testing of network designs. A representation of a network model is extended to store additional information usable to configure a test process. Based on the model information, test code is automatically generated and utilized in automated testing of the network design. Each entity in a network graph is associated with attributes that include information about the entity and configuration templates that are pertinent to a relationship between the entity and one or more entities. Each relationship is also associated with one or more attributes including verification templates, whether the relationship is testable, and a relative dependency order of the relationship. Topology information is derived from a selected network model or sub-model. Test code is generated for each relationship using its configuration map and verification map. A code generator is implemented to generate code for a specified test framework.

KEYWORDS

- Network testing
- Network design
- Network model
- Network graph
- Design validation
- Test automation

BACKGROUND

Network design and modeling ensures that a network such as a communications network fulfills the requirements of the operator and users of the network. A common practice is to model the network design prior to its deployment for purposes of automation. During the design process, a network is commonly modeled as a graph that includes entities such as network components, network devices, etc. and relationships between the entities. Upon completion of the design phase, the network is validated by testing different portions of the network to ensure that the design functions as per specification. Such validation can be performed either as a manual process or by utilizing an automated test process. During production and deployment, a network design can sometimes undergo changes from an originally designed network configuration. This can lead to insufficient or inaccurate test coverage.

DESCRIPTION

This disclosure describes techniques for automated testing of network designs. Per techniques of this disclosure, a representation of a network model is extended to store additional information that can be utilized to configure a test process. Based on the model information, test code is automatically generated and utilized in automated testing of the network design. Since the tests obtain the configuration information from the network design, the test process automatically inherits accurate information from the network model even when the model is subsequently updated during deployment.

The network is represented as a graph that utilizes nodes to represent entities and edges to represent relationships between the entities. For example, an entity can be any network element, e.g., network device, port etc. Relationships can be defined between related entities to specify a

nature of the relationship. For example, a network device and its port may be related via a "contains" relationship that specifies that a network device contains its port.

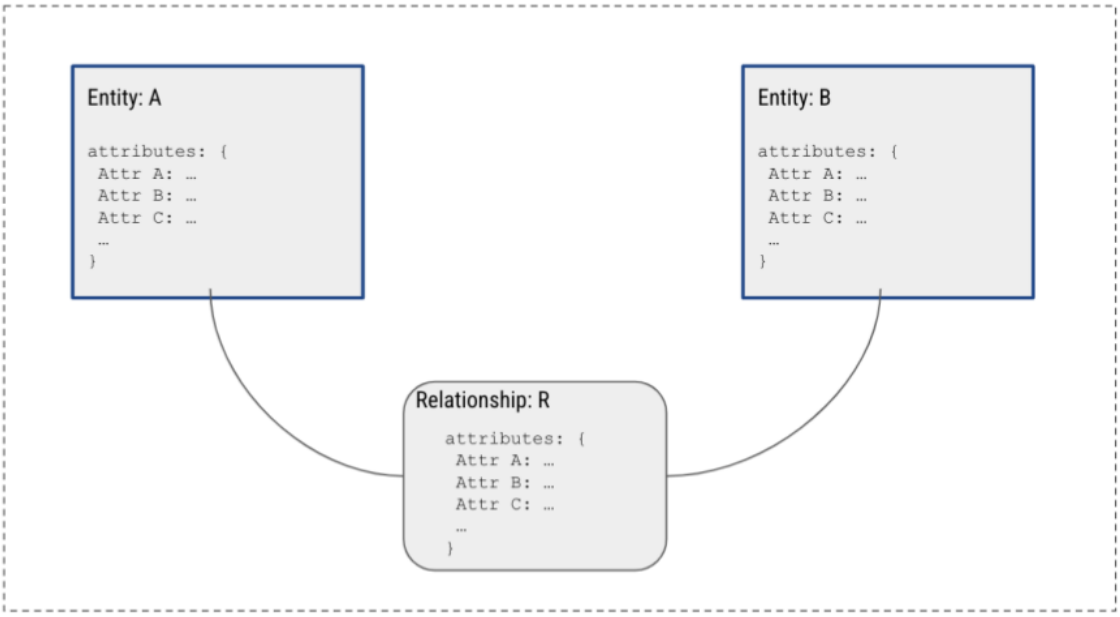

Fig. 1: A network graph is used to represent entities and their relationships

Fig. 1 depicts an example network graph that includes two entities - A and B that share a relationship R. Relationships can be unidirectional or bidirectional. Each entity is associated with one or more attributes that include information about the entity and configuration templates that are pertinent to a relationship between one or more entities. The configuration templates are modeled using a vendor neutral method such as OpenConfig.

Each relationship between entities is also associated with one or more attributes. The relationship attributes include information about the relationship such as:

- One or more verification templates. A verification template verifies the state of the relationship relative to an entity.
- Whether the relationship is testable or not.

- A relative dependency of the relationship specified as an order of the relationship. There may be more than one relationship between entities. In this situation, relationships are ordered to denote dependency.
- A configuration map of <Entity, Configuration Template> that establishes the relationship.
- A verification map of <Entity, Verification Template> that verifies the relationships on an entity.

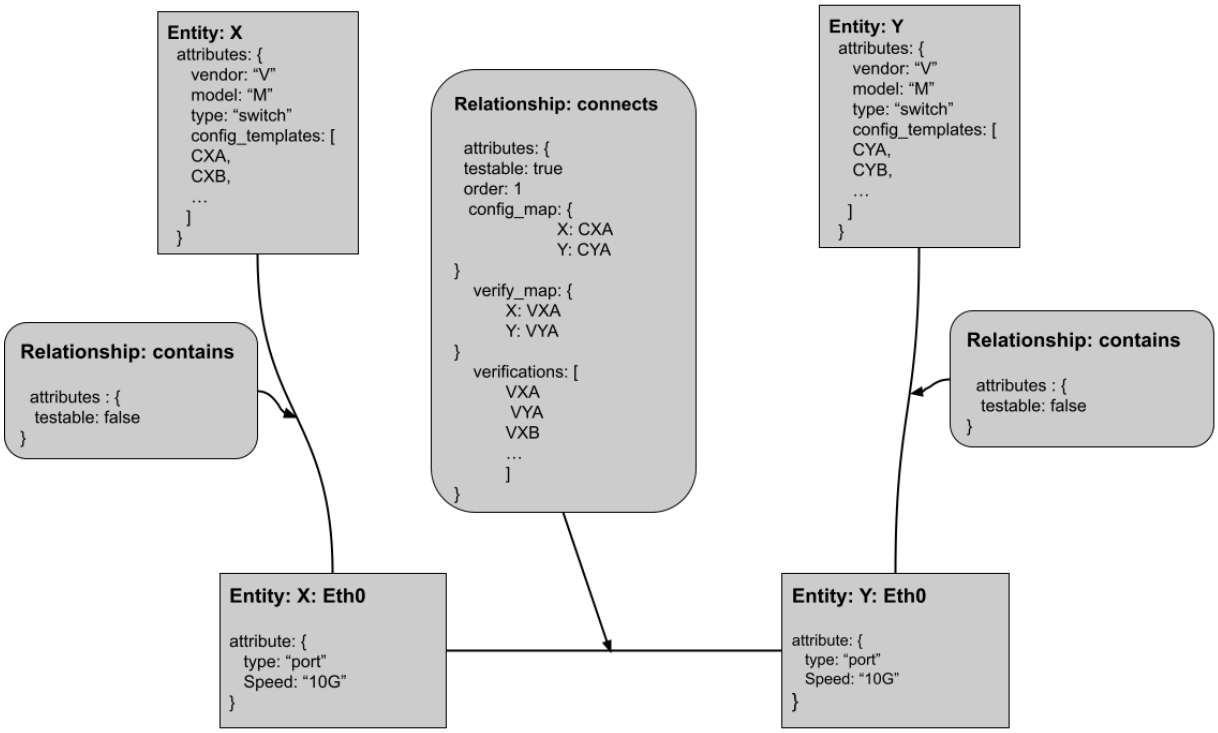

Fig. 2: Example entities and relationships in a network model

Fig. 2 depicts example entities and relationships corresponding to a network that includes two switches and an interconnect between the switches. As depicted in Fig. 2, an entity for a switch (X) has a "contains" relationship with its port (X: Eth0). This relationship between the switch and its port is marked as non-testable. However, two ports corresponding to different switches have a "connects" relationship between them which is marked as testable. The attributes

of the relationship specify information about the configuration templates to be applied to the entities and the verification steps.

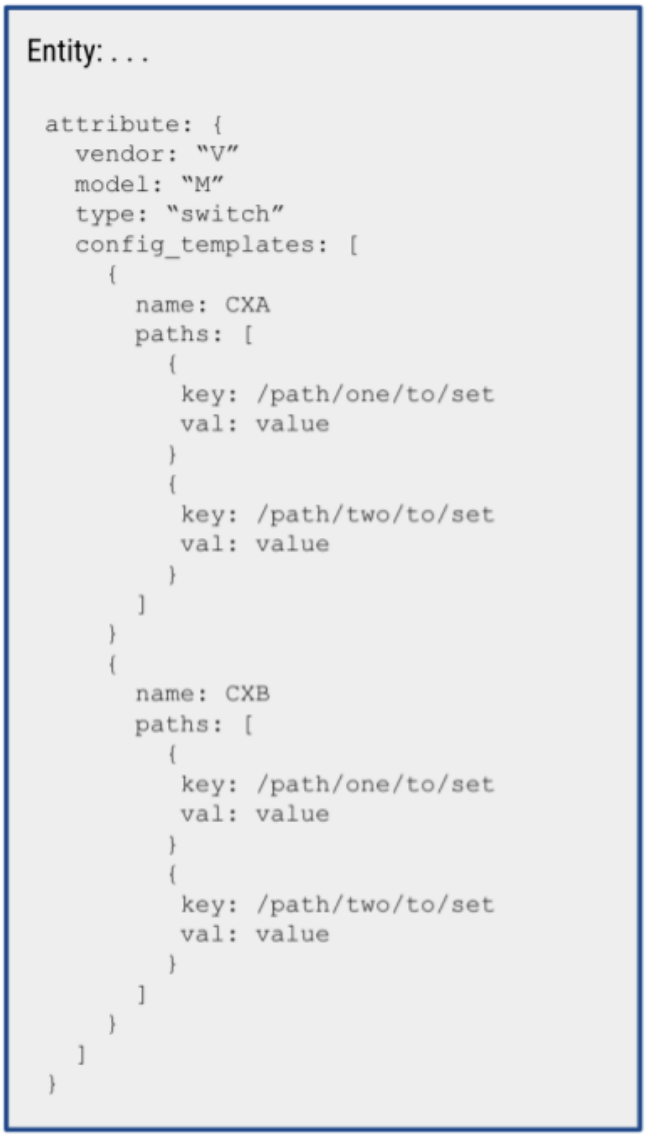

Fig. 3: Example entity attributes

Fig. 3 depicts an example of a detailed view of attributes for an entity. As depicted in Fig. 3, the attributes include details about the vendor, model number, type of entity, as well as configuration templates that are specified as key value pairs.

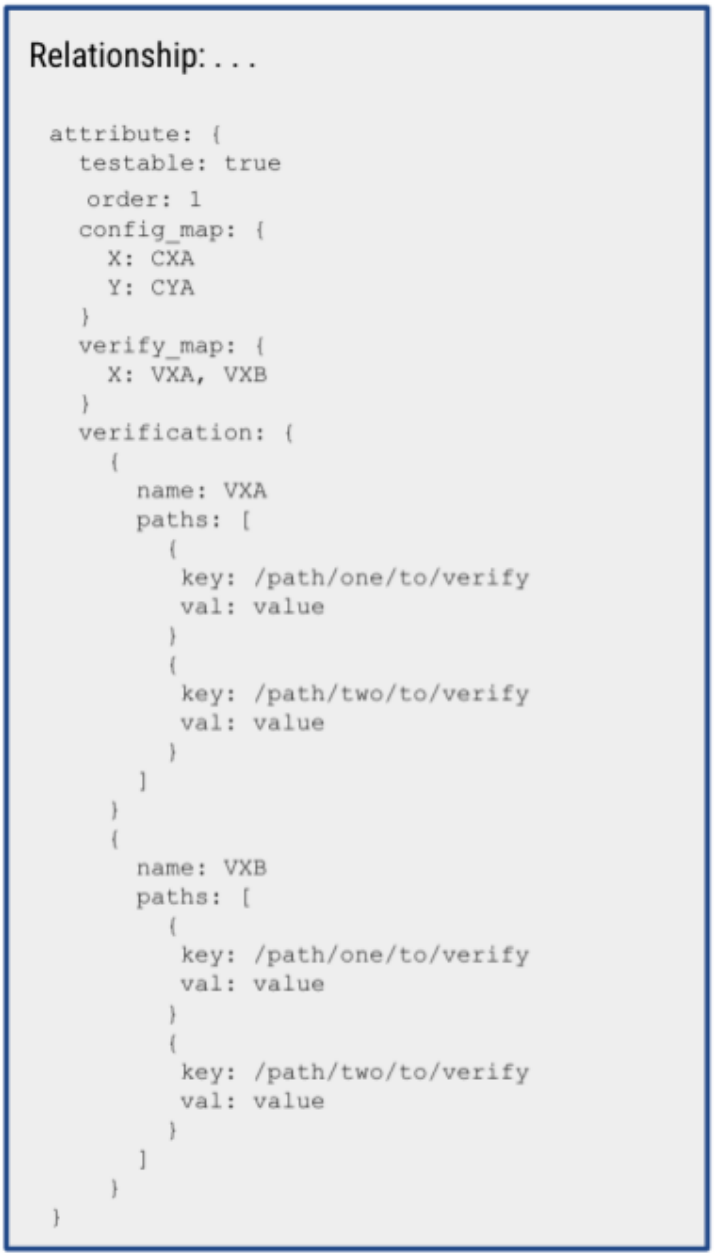

Fig. 4: Example relationship attributes

Fig. 4 depicts an example of a detailed view of attributes for a relationship. As depicted in Fig. 4, attributes for the relationship include whether it is testable, an order that specifies a dependency hierarchy, a configuration map, and verification maps. During testing, test scope can include either the entire network model or portions of the entire network model, e.g., a sub-

model. Topology information is derived from a selected model or sub-model and test code is generated for each relationship using its configuration map and verification map. A code generator is implemented to generate code for a specified test framework.

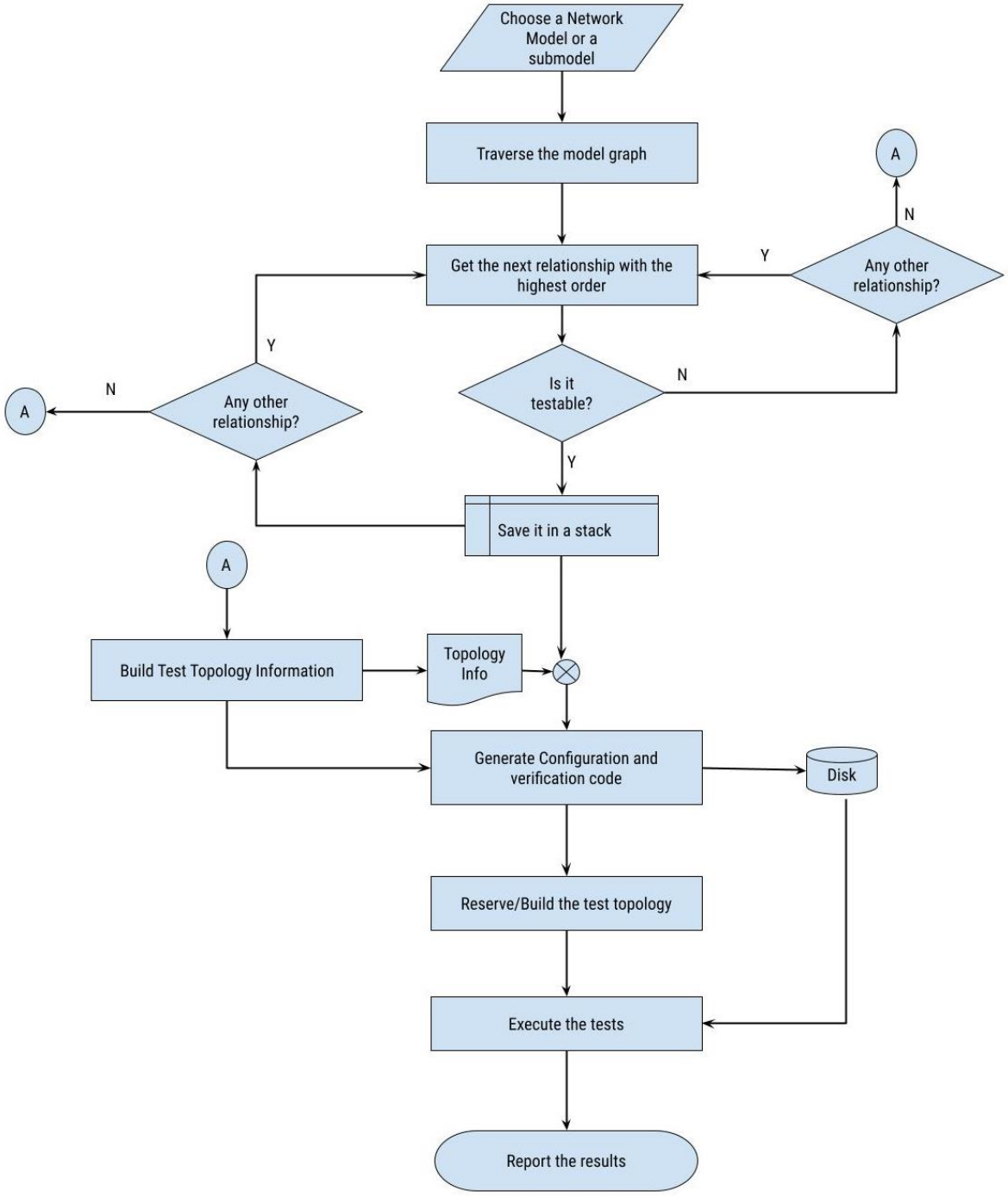

Fig. 5: Workflow for automated verification of a network model

Fig. 5 depicts an example workflow for automated verification of a network model, per techniques of this disclosure. A network model or sub-model is selected. The model or sub-model graph is traversed. A relationship with the highest order (of dependency) is obtained. It is determined whether the relationship is testable. If the relationship is testable, the relationship is saved in a stack.

This process is repeated until relationships of all orders (from highest to lowest) are evaluated. Once all relationships have been evaluated for testability, test topology information is obtained. Configuration and verification code is automatically generated based on configuration and verification templates. The test topology is reserved (built) and then executed. Based on test execution, the results are reported.

Techniques of this disclosure can be utilized to automate testing of network designs based on a network model. Since the test process obtains configuration information directly from the network design, it provides greater accuracy and superior test coverage.

CONCLUSION

This disclosure describes techniques for automated testing of network designs. A representation of a network model is extended to store additional information usable to configure a test process. Based on the model information, test code is automatically generated and utilized in automated testing of the network design. Each entity in a network graph is associated with attributes that include information about the entity and configuration templates that are pertinent to a relationship between the entity and one or more entities. Each relationship is also associated with one or more attributes including verification templates, whether the relationship is testable, and a relative dependency order of the relationship. Topology information is derived from a selected network model or sub-model. Test code is generated for each relationship using its

configuration map and verification map. A code generator is implemented to generate code for a specified test framework.

REFERENCES

1. “OPENCONFIG: Vendor-neutral, model-driven network management designed by users,” https://www.openconfig.net/ accessed on August 16, 2022.