Topology Generation for Protocol Design and Testing

Ganesha Bhaskara*, Sandeep Gupta*, Ahmed Helmy$
ganesha@bhaskara.org, sandeep@poisson.usc.edu, helmy@cise.ufl.edu
* University of Southern California, $ University of Florida

I Introduction and Problem Statement

In Internet Protocol based networks, end to end protocol testing is necessary but not sufficient due to the large scale use of intermediate nodes that may not adhere to the end to end principle. Network address translators, load balancers, virtual private network tunnels are some of the examples that break the end to end principle. While testing, differentiating between topologies that may or may not trigger a problematic behavior is non trivial and is often left to the protocol designer's discretion. If the set of all topologies can be classified into *protocol specific* sets, where each is characterized by the necessary and sufficient topology conditions, the number of topologies over which the protocol must be tested may be significantly reduced. However, a direct approach to identify such precise necessary and sufficient conditions of topologies will incur very high computation complexity as it will not only require an exhaustive exploration of the space of all possible topologies, but also precisely solving the general reachability problem for each instance of the topology. Existing work [1, 3] either implicitly assumes broadcast topology or expects the designer to explicitly provide the topology configurations as an input [2]. [4] reduces the problem of testing a system with n identical processes to testing a system of k processes, where k < n. This approach only works for processes on a broadcast medium, which is only one form of connectivity in IP networks.

In this paper, we present a new algorithm COMPRESS (COMPressed REpresentation of State Space for topology generation), to generate necessary conditions that all static IP based topologies must satisfy for the target behavior to occur, without resorting to explicit topology enumeration. We accomplish this by developing a complete but compressed representation of the state space of the protocols and developing a partial topology representation with richer expressiveness that can represent a potentially infinite set of topologies in a compact manner. Using case studies, we show that COMPRESS is a useful tool for an interesting class of protocols that also includes DHCP, ARP, RARP, Zero-Conf, client server protocols and so on. This work complements many of the existing formal methods as it systematizes the problem of topology generation.

II Solution Approach

COMPRESS takes as inputs, ***(a)*** network protocols modeled as extended finite state machines (eFSM) described using state transition tables, ***(b)*** topology rules including topology model and packet delivery rules, and ***(c)*** a given target behavior of the protocol represented as a sequence of eFSM state transitions.

A topology is represented as a graph, where vertices represent nodes and edges represent connectivity or links between the nodes. The topology rules specify constraints on types and cardinalities of nodes that can be present in the adjacency list of the different types of nodes. Unless explicitly forbidden, new instance of vertices (nodes) and edges (links) may be added to a topology graph or existing instances of vertices and edges may be merged. Thus, in a fully specified topology, the adjacency list constraints are specified such that no new vertices or edges can be added or no existing vertices or edges can be deleted or merged. Thus, *a partially specified topology represents a potentially infinite set of fully specified instances of topologies that satisfy the specified conditions*. We use this representation to capture necessary topology conditions. For each message destination type (e.g., MAC unicast, MAC Broacast, IP unicast, IP broadcast, IP multicast TCP etc), we model a finite set of partial topologies representing all ways in which a node sending a message and a node receiving a message using that message destination type, can be topologically connected. Thus, in any fully specified topology configuration, if a node sent a packet using a particular message destination type, then the connectivity between the node receiving that message and the node sending that message will be included in one of the sets of partial topologies.

The protocol behavior also known as the target behavior is represented as a sequence of eFSM state transitions. For example, the target behavior, “a node requesting an IP address and eventually receiving an address from the DHCP server”, specifies a behavior using two eFSM state transitions. Doing so enables us to specify a behavior of interest without knowing the exact number of nodes and their connectivity. Thus, target behavior is a set of structures, where each structure has the same form as a row in the eFSM state transition table, namely <consumed message, state transition, output message>.

COMPRESS is developed on the following key observations of the properties of the eFSM state transition table.

Property1: A bipartite graph with two classes of vertices denoting messages and eFSM state transitions respectively, can be constructed from the eFSM tables of all the protocols that interact with each other in the

system under study. This also happens to be a transitive closure graph (TCG) which contains every type of message and state transition that could occur on any fully specified instance of the topology.

Property 2: Consider a networked system *s,* with a given number of eFSMs connected to a broadcast medium like LAN. Consider a path *p* in the evolution of state of *s* that contains the target behavior <consumed message 1, state transition 1, output message 1> eventually leading to <consumed message 2, state transition 2, output message 2>. The path *p* can be decomposed into a tree made up of its component eFSM state transitions. At least one branch, say b_1, in that tree will represent the target behavior. Since eFSM specification describes all the transitive relationships between the messages and state transitions, there will always exists at least one path, say t_1, in the TCG between the eFSM state transitions types specifying the target behavior, whose intermediate state transition types represent a partial order on the set of all eFSM state transition instances in b_1.

If all the static information in t_1 is used to instantiate eFSMs, then the instantiated system will be partially specified, i.e., it will *always* contain a subset of the eFSM instance in *s*. Thus, for a given target behavior, a partially specified subset of *any* instance of the system with broadcast connectivity can be constructed from the TCG. Thus, the TCG represents a compressed but inclusive sequence of evolution of state space of the set of all instance of the networked system with broadcast connectivity.

We augment the TCG with node to node connectivity information obtained from message destination type information embedded in the eFSM state transition tables, to make it a tripartite graph where the three classes of vertices correspond to message, eFSM state transitions, and partial topologies per message destination type. Since the set of partial topologies per message destination type includes all ways in which a node sending a message and a node receiving that message can be connected, this version of the TCG exhibits the same state transition inclusion property as before, but for all topology connectivity configurations. Thus, using the augmented TCG, a partially specified topology can be constructed for target behaviors that occur on any instance of a fully specified topology. The TCG may contain multiple paths that represent the target behavior. Since it is not possible to determine which one of the paths represents the true partial order of eFSM state transitions, we extract the static topology properties from all paths representing the target behavior in the TCG and the intersection of the partial topologies extracted from all paths represents the necessary topology conditions.

We have used this methodology to generate topology conditions for 7 scenarios of the multicast based micro-mobility protocol and 5 scenarios of enterprise wide resource discovery protocol based on an extended version of ZeroConf protocol. We were able to generate necessary conditions that are close to sufficient conditions at a low overall complexity for most of the scenarios that we have studied. The occurrence of the behaviors on fully specified topologies generated using necessary conditions have been verified using state space analysis. Though the topology sizes identified were small, their configurations were non-intuitive.

III Contributions and Future Work

In general testing eFSM based systems for reachability is computationally expensive. Further testing a system using iterative approach for topology can neither guarantee coverage nor completeness of testing. In this work, we have developed a methodology that can perform topology generation without explicit topology enumeration and without solving the general reachability problem. We believe that this is the first work that deals with behavior based necessary topology conditions generation. The necessary conditions can also be used to judge the severity of the problem as the necessary conditions may be able to estimate the portion of the topology space on which the problematic behavior may occur. Though the worst case complexity of the algorithm is exponential, experimental results show that this methodology works with a low overall complexity for many different protocols whose behaviors are dependent on topology configurations. We have used COMPRESS on multiple classes of protocols and have generated necessary topology conditions for many scenarios at a low overall complexity.

We plan to augment the topology generation algorithm to be more efficient. We are performing more case studies to highlight the utility and limitations of the methodology.

IV References

[1] Dietrich, F., Hubaux, J.-P., “Formal methods for communication services: meeting the industry expectations,” pp. 99-120., Vol. 38, No. 1, *Computer Networks*, Jan. 2002,

[2] Ismaïl Berrada, et. al., “Testing Communicating Systems: a Model, a Methodology, and a Tool”, Page 111-128, Volume 3502/2005, Lecture Notes in Computer Science, ISBN978-3-540-26054-7, 2005.

[3] R. Lai, “A survey of communication protocol testing”, Pages: 21 - 46 , Volume 62, Issue 1, Journal of Systems and Software, ISSN:0164-1212, Elsevier Science Inc, 2002

[4] E. Allen Emerson, Thomas Wahl, “Efficient Reduction Techniques for Systems with Many Components”, Symposium on Formal Methods (SBMF), Recife/Brazil, 2004.