Journal of Network and Computer Applications 32 (2009) 184–214

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

A virtual environment for network testing

Wade A. Fagen [a], João W. Cangussu [b,*], Ram Dantu [c]

[a] University of Illinois at Urbana-Champaign, USA
[b] University of Texas at Dallas, TX 75083, USA
[c] University of North Texas, USA

ARTICLE INFO

Article history:
Received 23 April 2007
Received in revised form
11 March 2008
Accepted 24 March 2008

Keywords:
Virtual network
Software testing
Re-configuration

ABSTRACT

The testing of network-based solutions demands a series of tedious tasks such as the deployment of the solution at different nodes and the configuration of different topologies. The manual execution of these tasks is very time consuming and a configurable environment to facilitate these tasks and consequently improve testing performance is desired. In this paper a virtual network environment that can be easily re-configured is presented to address this problem. The environment has been evaluated by a series of case studies: one dealing with the deployment and containment of a worm propagation attack and one dealing with detecting a denial of service attack. Three smaller case studies have also been developed. The results are a clear indication of the flexibility and usefulness of the virtual network environment.

1. Introduction

In nearly every networking research endeavor, it is critically important to test at least some portion of the network protocol or solution introduced. For this task, some rely on simulation environments provided by tools such as ns-2 (Fall and Varadhan, 2007) while others have complex arrays of machines dedicated simply to the task of testing network software and protocols (Lundgren et al., 2002). While both of these approaches have their benefits, currently not much middle ground solutions between a simulation environment and a full fledged dedicated network exist.

One of the greatest advantages of a simulation environment is that every environmental factor is virtually limitless. Since the number of nodes in the simulated network only increase the amount of

* Corresponding author. Tel.: +1 972 883 2193; fax: +1 972 883 2349.
E-mail addresses: wfagen2@uiuc.edu (W.A. Fagen), cangussu@utdallas.edu (J.W. Cangussu), rdantu@unt.edu (R. Dantu).

1084-8045/$ - see front matter
doi:10.1016/j.jnca.2008.03.008

processing time required to complete the simulation, the task of setting up a 10,000 node network becomes trivial (Fall and Varadhan, 2007). Moreover, simulation environments allow for the strict control of a network—allowing for the simulation to place a designed algorithm in guaranteed worst case scenarios such as high packet loss or slow response times. However, since simulation environments are not run in real-time, the execution of simulated networks must take place a window of time at a time. Therefore, the amount of time between a node receiving a request and responding to that request is more often than not strictly defined. In the cases where the response time is not strictly defined, the response time is defined by the result of a random number generator or some other similar method of generation. As such, a simulation will never be able to fully achieve an asynchronous network model as present in real networks.

On the other hand, physical networks provide the benefits of having a completely asynchronous network with individual processes allowing for the each individual computer to respond in real-time to incoming messages (Buyya et al., 2002; Macedonia et al., 1994). However, with the use of physical computers, each computer must be individually configured. The application being tested must be deployed to each computer, be able to ran at a specific time, and the amount of nodes in the network is limited to the number of physical computers available. Therefore, it would be unrealistic in the vast majority of cases for a 10,000 machine physical network to be deployed to test out a given algorithm.

A solution which rests between the simulation environment and a full fledged physical network, in the "middle ground" of networking, is able to achieve an asynchronous network model due to the fact that the application being tested is deployed to each individual computer as well as not having the one-to-one limitation between the number of physical computers and the number of individual applications being tested. As such, it may be realistic for 500 machines to be obtained for the testing of a new algorithm. With twenty instances of the algorithm running on each machine, a 10,000 node "virtual network" can be created with many of the strengths of a full physical network. However, many network virtual nodes may reside inside a single physical node and the results of the use of such solution cannot be considered the same as if tested in a real network environment.

Of the widely available solutions which try to fill this middle ground, every solution which was found either required a specific proprietary programming language, a specific operating system running on each client machine in the cluster of networked computers, or some other limiting factor which prevented an easy transition between network researching programs, protocols, or network topologies.

After exploring the currently available solutions, a criterion was developed to determine the robustness and usability of a solution which allowed for the distribution of a program or component to individual computers to allow for network testing. The criteria consider four key areas as defined below.

- *Interoperability*: Can the given solution natively run across a wide variety of operating systems? If so, can the given solution natively run the network component being tested on all of the supporting operating systems? If so, can this be done without having the need to compile a specific version for each operating system?
- *System usability*: Can the given solution successfully run as a user (non-root in Linux/UNIX-based systems, non-Administrator in Windows-based systems) without degrading the functionality of the solution? Often at universities and research labs, numerous computers are available for public use where each user is given some limited-privileged user account login. If such a login is sufficient to run the given solution, the availability of computers within an institution is greatly enhanced.
- *Flexibility*: Can numerous tests be scheduled within the given solution? If so, can tests be scheduled while current tests are in progress or must all tests be scheduled at the beginning? Can a test be preempted by another test at a user's request? Can all of this be done without needing to restart or re-instantiate the given solution?
- *Adaptability*: Can different networking components be ran throughout the series of tests? Can the topology differ from the physical topology connecting the computers? If so, can this differing topology be reconfigured throughout the series of tests?

Through research and evaluation of widely available solutions, no solution provided adequate results based on the criteria above. Solutions deployed currently are nearly all platform and language