Load Balancer Discovery for the Data Center: a Holistic Approach

Joel Branch, Michael Nidd, Rüdiger Rissmann
IBM Research, Watson and Zürich Labs
branchj@us.ibm.com, {mni, rri}@zurich.ibm.com

Abstract— **Data center migration relies heavily upon the ability to detect critical IT components in the original environment. Not all components are easy to detect using traditional discovery tools. For instance, load balancers are designed to be transparent to network traffic, making them invisible to normal scans. Also, the discovery task must be minimally invasive, since the IT environment to be migrated is usually in a production state until the relocation is complete. Studying the configuration of discovered host machines can reveal the presence of the network appliances they use, allowing a "short list" of probable appliance locations to be made, even if such appliances do not themselves respond to traditional network scans.**

This paper describes a collection of heuristics for detecting the presence of network devices in data centers, with a specific focus on detecting load balancers. Our approach lies in exploiting (1) host-based network data, which is often collected during data center migrations, and (2) knowledge of how load balancer appliances and managed servers are conventionally configured in an enterprise network. The effectiveness of our techniques is evaluated using data from an IBM data center migration project.

Keywords: *network discovery; appliance discovery; load balancer; data center*

I. Introduction

Some network appliances, such as firewalls and server load balancers, are very difficult to discover automatically when they are properly deployed. Such devices are designed to operate discretely, simply relaying or redirecting requests and responses. However, scenarios such as data center migrations, require these devices to be identified. Here, it is important to first discover *critical* devices and associated configurations in the source network so that the target (e.g., cloud) network can be configured to reflect the original's functionality. Firewalls and load balancers have become increasingly important to enterprises due to their security- and performance-centric operations; hence their detection is especially important. Network discovery tools, some of which may detect firewalls and load balancers, do exist. However, their usage in data center migrations may be restricted for various reasons.

Data center migration supports many larger IT operations (e.g., outsourcing, transformation) and comprises the movement of an enterprise's selected IT assets to a new environment [4][5][15]. Generally, data center migration is disruptive and labor-intensive; hence, there is a strong motivation to complete the initial discovery phase quickly. While deploying full-featured "probe-based" network discovery tools (e.g., IBM® Tivoli® Network Manager and OPNET® NetMapper®) in this phase would deliver comprehensive results, their usage may be prevented due to concerns such as network security, environmental stability, and licensing cost.

This paper describes a collection of heuristic-based techniques for detecting the presence of network devices, namely server load balancers, in data centers, avoiding traditional high-overhead probe-based approaches. The crux of our approach lies in exploiting (1) host-based network data, which comprises conventionally and more effortlessly collected information in data center migrations, and (2) knowledge of how load balancers and managed servers are conventionally configured in an enterprise network. We also present an analysis that shows the potential of our heuristics, laying the groundwork for further research in this area. The remainder of the paper is arranged as follows: Section II provides a detailed background of our approach; Section III describes our discovery heuristics; Section IV presents the evaluation; and conclusions are drawn in Section V.

II. Proposed Approach

Existing work in data center network discovery falls mainly into either of two categories: (1) being the first step in the deployment of a comprehensive monitoring system, or (2) being part of a penetration test. In (1), the goal is to establish broad coverage first, then develop complete coverage over time. In (2), the goal is to gain specific knowledge of potential vulnerabilities without calling attention to the network scan. In our case, time restrictions and the fact that we do not need to establish a lasting infrastructure (due to often requested network redesigns) render the first group of tools inappropriate. Also, penetration testing is most useful when there is a particular target in mind, rather then for scanning the whole environment. Overall, as we demonstrate in this paper, having permission to request SNMP [3] or even remote login access to specific hardware in a data center migration engagement means that we are not so limited in our options as a penetration test team would be. The challenge is reduced to one of knowing the potential location of these devices, so we can ask for access and collect further details directly.

Given the afore-mentioned limitations, we propose a load balancer detection approach that exploits network data obtained from hosts/servers in the data center, since one typically knows of and has access to most (or all) of such devices in migration engagements. We look for patterns that are indicative of well-known load balancer-oriented network configurations, such as may be observed in local network interface card configurations and ARP caches. We also look for host data that exhibits generally unusual device configurations, such as similarities between hosts' file systems and naming properties that might suggest cluster membership.

The environment in which we have developed our solution is based on the IBM Migration Factory service to facilitate IT infrastructure discovery for data center migrations [5]. The initial discovery task uses a combination of remotely managed scripts from IBM TADDM (Tivoli Application Dependency Discovery Manager) and local scripts that are executed directly by the enterprise's server administrators. The information is filtered and placed into a database, from which various reports (e.g., server dependency information, utilization, and the results of our heuristics) are generated. Our heuristics are implemented either directly by the IBM Tivoli Common Reporting engine or in Java programs. The results of the heuristics are sent to at least one human operator who can confirm the existence and location of devices as suggested by the heuristics and then enter more detailed device information in the database for use by subsequent migrations phases.

Most of the data that we use is obtained via local and remote scripts or SNMP. The local and remote scripts are primarily used to gather configuration information from servers. For instance, Galapagos [11] is a suite of local scripts we employ that runs utilities like Unix `ifconfig` and `lsof`, then archives the results along with copies of particular configuration files. TADDM uses access credentials (login or SNMP) for some set of servers, so it can connect from the network and do much the same thing as Galapagos.

In more interactive scenarios, direct connections to the network APIs of installed middleware and "fingerprinting" systems (e.g., NMAP [9]) can improve load balancer detection and include other devices such as firewalls. The long delay between running scans and the lack of consistent live connections to the subject environment from the analysis environment are among the reasons for studying alternatives to the direct scanning techniques such as those using NMAP.

III. Network Discovery Heuristics

Load balancers frequently require distinctive environmental configurations. We exploit the knowledge of some of these configurations in analyzing network data collected from host machines to detect patterns that could indicate the existence of load balancers and managed clusters. The nature of these patterns ranges from the configuration of single devices (including server loopback interface configurations) to the distribution of properties across multiple devices (such as multiple hosts on a single subnet having different default gateways).

A. Single-device heuristics

1) Analyzing server loopback network interfaces

One conventional load balancer deployment configuration is *direct routing*, as illustrated in Figure 1. Here a load balancer accepts requests from clients on a virtual IP address ("VIP" in Figure 1) and forwards them to servers on the same subnet by changing the destination MAC address. The servers adopt the IP address of the service so that when they send responses directly to the client, the client assumes that the response came from the originally requested service (due to the source IP of the packet being the service's IP). Here, in order to enforce correct network behavior, the loopback interfaces of the clients ("lb IP" in Figure 1) must be configured as the VIP of the load balancer device. Hence, detecting loopback interfaces that are not "127. 0.0.1" may reveal load balancer activity. There could be other reasons why such a loopback configuration is used, so this technique may give false positives. To further improve confidence, we need to add other patterns, possibly looking for similar host/DNS names, comparing the list of running services/open ports, or verifying that the servers reside on the same subnet.

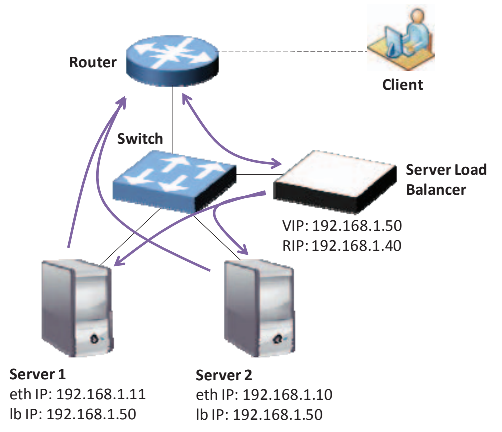

Figure 1. Load balanced servers configured in a direct routing design.

2) Analyzing gateway MAC addresses

Load balancers are traditionally assigned virtual IP and in some cases, *self-generated* MAC addresses to support, among other functions, redundancy for failover configurations [8]. Hence, using such information can be helpful in identifying actual load balancer machines, as opposed to the previous heuristic which mainly aims to identify servers connected to load balancers. Since there are no (reliable) means of identifying virtual IP addresses, we focus on identifying self-generated MAC addresses as a way of identifying potential load balancers.

Self-generated MAC addresses differ from universally administered addresses, i.e., those containing organizationally unique identifiers, mainly by the pattern of the first two significant bytes of the MAC. For some load balancers, this pattern will be "02-bf" (or "01-bf" or "03-bf"). Here, the first byte indicates the type of load balancer configuration (IGMP, unicast, or multicast) and the second is a placeholder for the priority of the load balanced server handling a client request[1]. While this MAC pattern should identify a device as being a load balancer, it is not guaranteed that the pattern will always be used. However, as will be understood by reading the next subsection, using this heuristic in combination with some multi-device heuristics could help increase the confidence level of load balancer detection.

B. Multi-device heuristics

While analyzing data about single hosts can be helpful, analyzing data across *several* machines may yield a higher

1 A priority value replaces "bf" in certain packet responses so as to enforce uniqueness in routers' tables when redundant load balancers are present.

probability of detection simply because load balancing usually involves multiple servers. We describe our multi-device heuristics below.

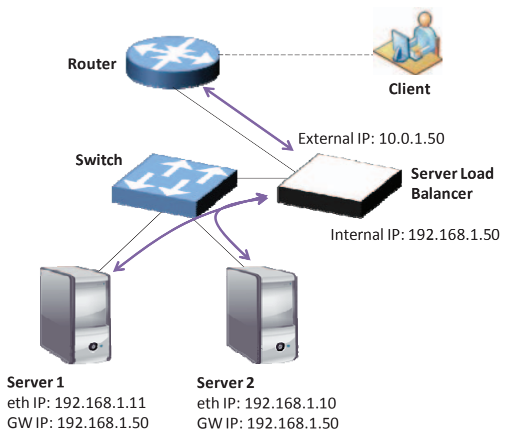

Figure 2. Load balanced servers connected through a NAT design.

1) Analyzing subnet gateway assignments

Analyzing the gateway assignments of a group of hosts in the same subnet can be helpful for predicting the presence of load balancers. Before describing the details of this heuristic, we first describe the premise on which it is supported.

Figure 2 illustrates a network configuration in which a server farm is connected to the load balancer via a layer 2 switch. The address of the service is an external virtual IP address of the load balancer. When the load balancer receives a request from a client, it uses *network address translation* (NAT) to translate the destination address to that of one of the servers ("eth IP" in Figure 2).

When the request reaches the servers, it will have the IP address of the server as its destination and the client IP address as its source address. Because of this address translation the server can not directly send the response packet back the client, since the client will expect a reply to come from the original VIP. Therefore, the server needs to send the response packet back to the load balancer, which will then reverse the translation and forward the response on to the client. To achieve this routing pattern, the load balancer needs to be configured as the default gateway for its managed servers. Other servers on the same subnet that are not load balanced would directly use the subnet router as default gateway. Hence, identifying subnets with more than one default gateway may indicate the use of load balancers.

2) Distance-based server clustering

In addition to analyzing hosts' connectivity features for hints of load balancer associations, we also take a more general approach of identifying servers that are likely to be in a cluster. In the implementation of this technique, we have defined metrics that indicate configuration similarities such as may be found between servers that provide the same service (via a load balancer). In the extreme case, server clusters consist of identical servers. In instances in which load balancing is used to increase the security and resiliency of services, the cluster might consist of different hardware and/or software products that provide the same service; but in most cases clusters will exhibit identical (or very similar) hardware and software to among other things keep administration as simple as possible. The distance metrics that we have used are based on the server name, and the open port list.

Using a weighted Levenshtein distance [10] between server names measures the number of editing operations required to change one name to another. Because we are looking for name groupings, not typing errors, inversions are counted as two operations (rather than one), and substitutions of a digit for another digit are counted only half as significant as insertions, deletions, or other substitutions. An efficient algorithm for this calculation was published by Wagner and Fischer [14].

The distance between open port lists is calculated separately for the well-known ports (0-1023) and for the rest of the port numbers. The ratio of ports in common to total ports open on either machine in the test was weighted for each group (80/20 in favor of the general-purpose ports, since well-known ports are more likely to be shared between un-connected machines) to produce a value between zero and one hundred.

While the name and port list offer a good first pass in detecting load balanced server clusters, other properties that may better describe machine *functions* can also be used if needed. Other distance metrics that we have considered for future study include the following:

- Directory structure distance, or alternately, the distance between installed software;
- Database table (and cardinality) distance, especially if table sizes are large and few other (memory consuming) applications are installed.

IV. Evaluation

We evaluated the proposed heuristics using data collected during a real data center migration engagement for a large hospitality corporation. The data contained network configuration and other host properties (e.g., that describing software stack and file systems) collected from approximately 2200 servers. For this engagement, the location and configuration of load balancers in the network were all known. These configurations were used to produce a definitive answer regarding which servers in the environment were actually receiving traffic from a load balancer. This includes all possible configurations, including simple redirections (i.e., single-server clusters receiving traffic). Hence, we had a comprehensive data set by which to evaluate our heuristics.

A. Loopback Network Interfaces

Our data collection tools did obtain descriptions of the loopback interfaces of the hosts in the network sufficient to identify occurrences of load balanced machines configured in a direct routing configuration as described in Section IV.A.1. Because no production data to which we had access contained examples of load balancing within the same subnet (i.e., *MAC-layer load balancing)*, no full end-to-end tests were possible for this heuristic. However, references such as [13] do recommend MAC-layer load balancing as an industry best practice. Examples of its use include IBM WebSphere

Application Server load balancing [6][7], and Cisco IOS server load balancing (dispatched mode) [1][2], so this configuration is still worth discovering in practice.

A positive result of the evaluation was tha t although data was collected that would have detected MAC-layer load balancing, no false-positives were detected. This suggests that the heuristic should be reasonably reliable when it does detect a potential load-balanced machine.

B. *Gateway MAC Addresses*

In scanning available properties of the network interfaces in the test d ata for this paper, we did not find any self-administered MAC address patterns that were indicative of load balancer functionality. Ho wever, in preliminary evaluations using data collected from another data center migration engagement, we did confirm the effectiveness of this heuristic. S pecifically, using the MAC address pattern heuristic, we found several virtual servers identified as network load balancing filter d evices, as co nfirmed by an engineer working on the engagement.

C. *Analyzing Subnet Gateway Assignments*

Our test data identified subnet and gateway assignments for most of the hosts in the network; some data was missing due to gaps in the ongoing collection effort. For those hosts for which we had this information, we evaluated the subnet gateway assignment heuristic described in Section IV.A.2. Out of the 162 servers known to be receiving traffic from a load bala ncer, we identified 24 (or 15%) of them using this particular heuristic.

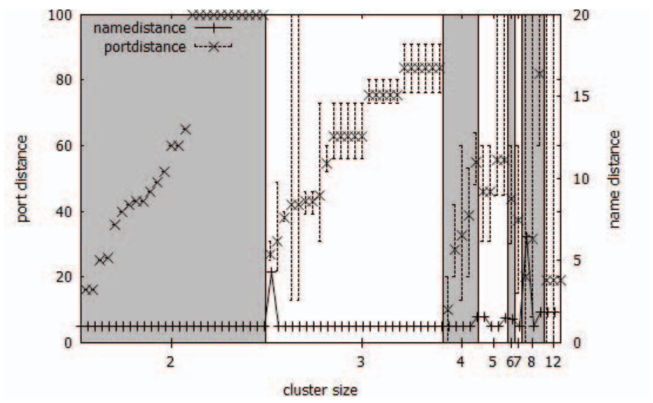

Figure 3. Evaluated Metric Values by Cluster Size.

D. *Distance-Based Server Clustering*

For the distance-based server clustering heuristic, we compared the average value for the m achine names metric and for the open ports metric b etween any pair of randomly chosen servers to the average value be tween servers that we knew to reside in a loa d-balanced cluster. Recall from Section IV.B.2 that the name distance metric ranges from 1 (a single digit-for-digit substitution) with no hard upper limit for very different names, and t he port similarity metric varies from 0 (i.e., no overlap in po rts) to 100 (i.e., all p orts are id entical). Th e results, showed a significant corre lation. The mean value for name metric distance between clustered pairs was 1.7, compared with a global average of 14.9, and t he similarity metric for the open port list was 37.6 for clustered pairs, with a global average of only 5.8.

In general, these results indicate that the metrics do make a useful contribution to determining whether two machines may be members of a single cluster. To further understand how this applies, consider the chart in Figure 3. Th is chart shows the average values for each of the two metrics for the 70 known clusters in the data set, sorted by cluster size (2, 3, 4, 5, 6, 7, 8, and 12 from left-to-right as i ndicated by alternating background shades; the data contained no clusters of size 9 through 11). For name distance metric, only the avera ge value is shown (on the scale shown o n the right), while the port distance metric includes a bar to s how the range from minimum to maxim um values in each c luster. The name similarity metric for small clusters (in this data set) is alm ost always 1, i ndicating only a digit change i n the names. For larger clusters, it is more common to have one or more outlier machines with very different names. Sim ilarly, for s mall clusters the port metric is usually very high, while larger clusters have more exceptions. This is stro ng evidence for the usefulness of these metrics as indicators that systems are members of a load balanced cluster.

V. Closing Remarks

In this paper, we present ed several load balancer detection heuristics for use i n data center migration engagements in which the use of traditional invasive network detection tools is not preferred. B y using production data from a migration engagement, we have sho wn that the heuristics presented here are effective in detecting several load balancer configurations, and that thes e configurations are relevant. Overall, our experience provides a unique contribution to the research and practitioner communities, and lays th e groundwork for lightweight detection of other devices.

As a resu lt of this study, we h ave identified several directions for continuing work. One is investigation of how these heuristics can be combined to potentially increase confidence and aut omation of detection. W e are currently investigating algorithms for best managing the workflow of heuristics execution, and w eighting the results of multiple heuristics in order to calculate a more comprehensive detection score. Another pursuit involves the development of heuri stics for other types of l oad balancer network configurations, including redundancy setups and m ore sophisticated uses of network/port address t ranslation. Such sol utions may involve collecting additional information from host machines. Further field testing will also be required for the refinement of existing heuristics, and we will b roaden the scope of our heuristics to cover devices such as firewalls, which share similar challenges regarding discovery in the context of data center migrations.

Acknowledgments

The authors thank the following for sharing their expertise in data center migrations and providing us with an evaluation test bed: R onda Caswell, Daniel Hernandez, Ying Huang, Venkata Kumar, Dennis Perpetua, and Birgit Pfitzmann.

References

[1] Cisco Systems, Inc. "C onfiguring Server Load Balancing," July 23, 2010.

[2] Cisco 7200 Series Routers documentation "Cisco IOS Server Load Balancing: Real Server Configuration," Document ID 10567, (http://www.cisco.com/en/US/products/hw/routers/ps341/products_tech_note09186a0080134735.shtml).

[3] J. Case, M. Fedor, M. Schoffstall, J. Davin, "A Simple Network Management Protocol (SNMP)," RFC 1157, May 1990. [Online]. Available: http://tools.ietf.org/html/rfc1157.

[4] M. Hajjat, X. Sun, Y-W. Sung, D. Maltz, S. Ra o, K. Sripanidkulchai, M. Tawarmalan i, "Cloudward Bound: Planning for the Beneficial Migration of E nterprise Applications to the Cloud," ACM SIGCOMM, pp. 243-254, 2010.

[5] IBM Migration Factory overview (http://www-03.ibm.com/systems/migratetoibm/factory/).

[6] IBM WebSphere Application Server v8.0 configuration guide "Edge Components, Version 8.0 > Load B alancer for IPv4 and IPv6 Administration Guide > C onfiguring Load Balancer > Configuring the Load Balancer machine" (http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp).

[7] IBM WebSphere Application Server v8.0 configuration guide "Edge Components, Version 8.0 > Load B alancer for IPv4 and IPv6 Administration Guide > Configuring Load Balancer > Configuring t he server machines > Aliasing the network interface card or loopback device" (http://publib.boulder.ibm.com/infocenter/wasinfo/v8r0/index.jsp).

[8] C. Kopparapu, "Load B alancing Servers, Firewalls, and Caches." New York, NY: John Wiley & Sons, 2002.

[9] G. Lyon, "Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning," Insecure.Com, December 2008.

[10] V. I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals," Soviet Physics Doklady, Vol. 10, No. 8., 1966, pp. 707-710.

[11] K. Magoutis, M. D evarakonda, N. Jouk ov, N.G. Vogl, "Galapagos: Model-driven Discovery of E nd-to-End Application-Storage Relationships in Distribu ted Systems," IBM Journal of Research and Development, No. 4/5, Vol. 52, July/Sept., 2008, pp. 367–377.

[12] C. Shaffer, "Identifying Load Balancers in Penetration Testing," Feb 2012. [Online]. Available: http://www.sans.org/reading_room/whitepapers/testing/identifying-load-balancers-penetration-testing_33313.

[13] Malcolm Turnbull, "Direct Routing aka. Direct Server Return on Windows 2008 using loopback adpter" for loadbalancer.org (http://blog.loadbalancer.org/direct-server-return-on-windows-2008-using-loopback-adpter/).

[14] R. A. Wagner and M. J. Fischer, "The String-to-String Correction Problem," Journal of the ACM, Vol. 21, No. 1, 1974, pp. 168-173.

[15] C. Ward, N. Aravamudan, K. Bhattacharya, K. Cheng, R. Filepp, R. Kearney, B. Peterson, L. Sch wartz, C.C. Young, "Workload Migration into Clouds – Ch allenges, Experiences, Opportunities," IEEE 3rd International Conference on Cloud Computing, pp. 164-171, 2010.